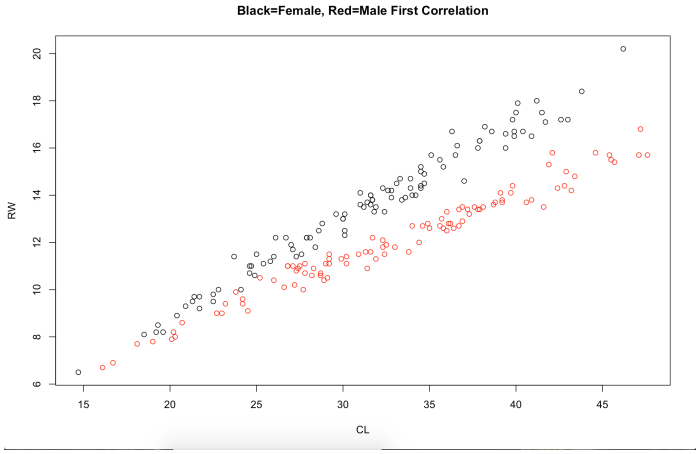
Computer lab 2 block 1

Alice Velander

December 8, 2019

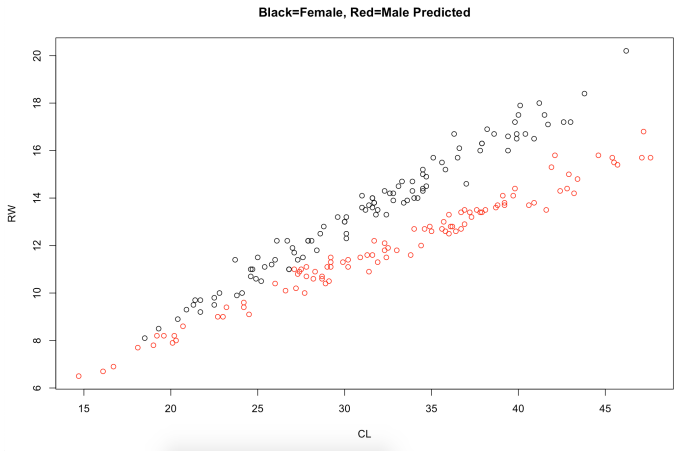
1 LDA and logistic regression

1. Make a scatterplot of carapace length (CL) versus rear width (RW) where observations are colored by Sex. Can we classify by linear discriminant analysis?



As seen in the scatter plot above, we can do an linear classification.

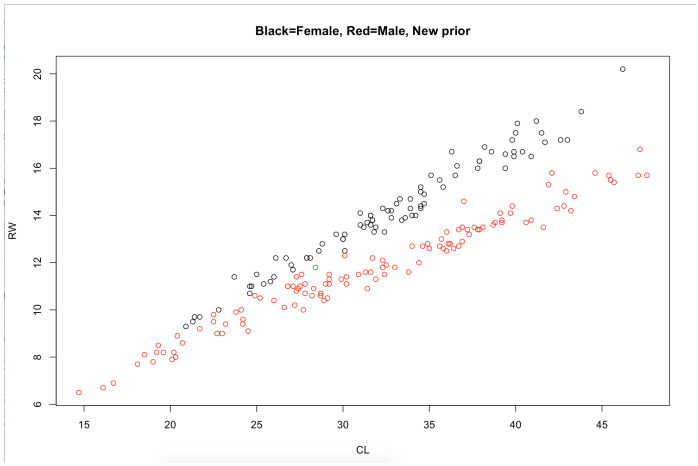
2. Make LDA analysis with target Sex and features CL and RW. Make a scatter plot of CL versus RW colored by the predicted Sex and compare it with the plot in step 1. Compute the misclassification error and comment on the quality of fit.



Misclassification rate = 0.035

Since we have very kind data that is easy to separate, we get a very similar plot as in step 1. We can also see that the prediction with LDA was good because of the low misclassification rate.

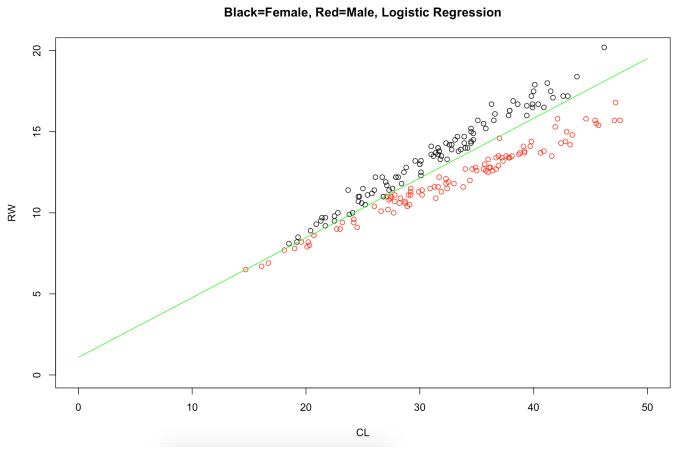
3. Repeat step 2 but use priors p(male)=0.9 and p(female)=0.1 instead. How did the classification result change and why?



Misclassification rate = 0.08

From the plots there are more males than females for the second prediction which is explained by the new prior. The misclassification rate is now slightly worse, making us think that the new prior does not help the prediction. This results in more females incorrectly predicted as males, and the likelikood of a male being falsely predicted as a female is low.

4. Make a similar kind of classification by logistic regression, plot the classified data and compute the misclassification error. Compare these results with the LDA results. Finally, report the equation of the decision boundary and draw the decision boundary in the plot of the classified data.



Misclassification rate = 0.035

With Logistic Regression we get the same misclassification rate as when we use LDA with prior=0.5.

The decision boundary line can be computed with

$$p(sex) = intercept + cRW \cdot RW + cCL \cdot CL \tag{1}$$

$$p(sex) = \begin{cases} 1/(e^{intercept+cRW \cdot RW+cCL \cdot CL}) \\ 0.5 \end{cases} \tag{2}$$

This gives us:

$$RW = -1 \cdot (intercept + cCL \cdot CL)/cRW \tag{3}$$

2 Analysis of Credit Scoring

2. Fit a decision tree to the training data by deviance and gini index. Report the missclassification rates for the training and test data.

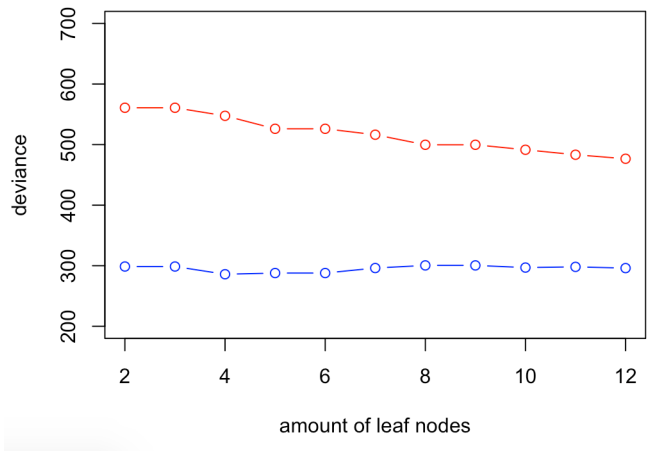
The following missclassification rates were given:

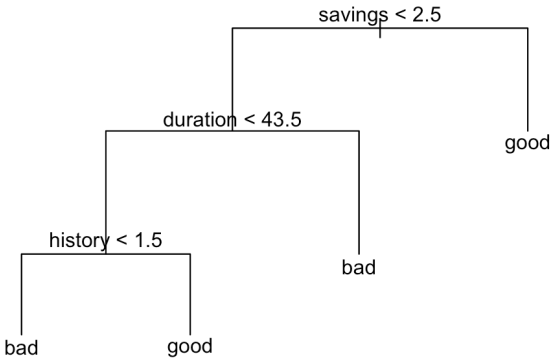
- Deviance on training data = 0.212

- Deviance on test data = 0.268
- Gini on training data = 0.242
- Gini on test data = 0.372

We can see that deviance gave us the best result with least errors on the test data.

3. Use training and validation sets to choose the optimal tree depth.





The blue line represents the results from the test data. As seen in the graph above, the optimal depth for the tree is 3 with four leaf nodes, where we have the minimum deviance.

The misclassification rate = 0.264. The chosen variables were duration, history and savings. From analysing the tree, if the person have savings below 2.5 we need to check the duration and thereafter if the duration is less than 43.5 we need to check the history to make the decision good or bad.

4. Use training data to perform classification using Naive Bayes and report the confusion matrices and misclassification rates for the training and for the test data.

Table 1: Confusionmatrix on Test

	0	1
0	46	30
1	49	125

Misclassification rate = 0.316

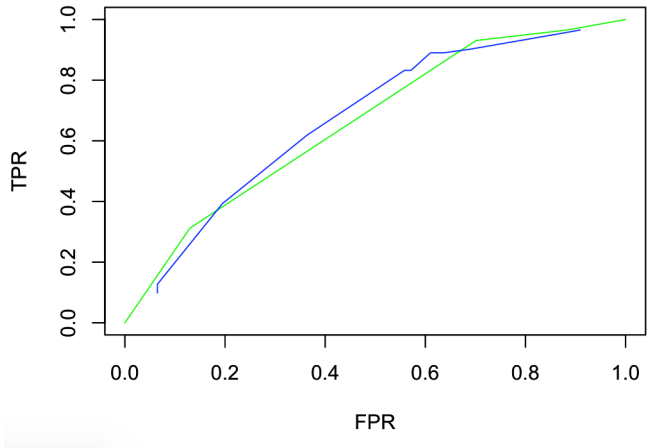
Table 2: Confusionmatrix on Training

	0	1
0	95	52
1	98	255

Misclassification rate = 0.3

Compared to tree classification (with split deviance) we get a higher misclassification rate when using Naive Bayes.

5. Plot the ROC curves from Naive Bayes and Tree.



Seen in the graph above the ROC curves are presented, green line representing tree and blue Naive Bayes. For lower and higher thresholds, tree classification gives us a better model compared to Naive Bayes.

6. From the given Loss Matrix the following applies:

$$p(\text{"}good\text{"}|x) > 10p(\text{"}bad\text{"}|x) \tag{4}$$

We get a higher misclassification rate, but this is because we have defined after the loss

Table 3: Confusionmatrix on Test

	0	1
0	71	5
1	122	52

Misclassification rate = 0.508

Table 4: Confusionmatrix on Training

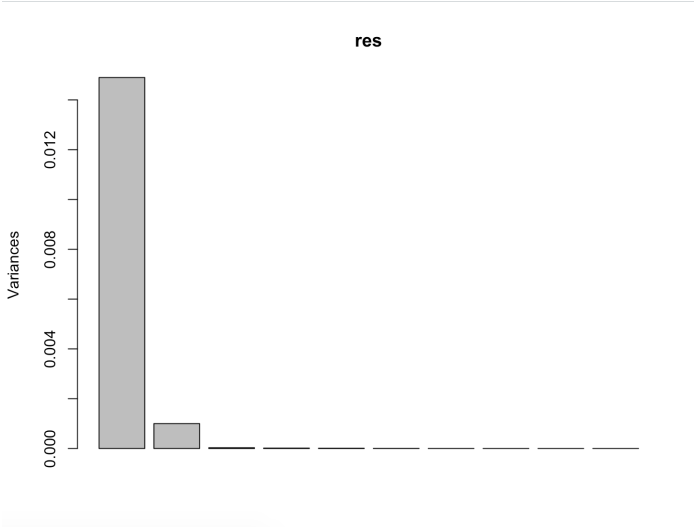
	0	1
0	137	10
1	263	90

Misclassification rate = 0.546

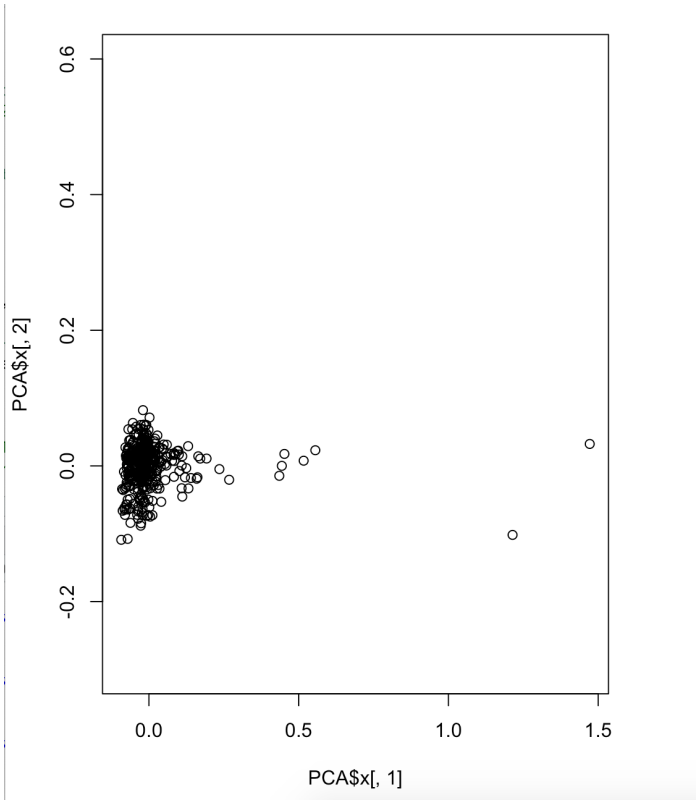
matrix, making sure we get a larger penalty if predicted bad. This does not need to say that it a worse model.

3 Principal Components

1. Conduct a standard PCA by using the feature space and provide a plot explaining how much variation is explained by each feature.

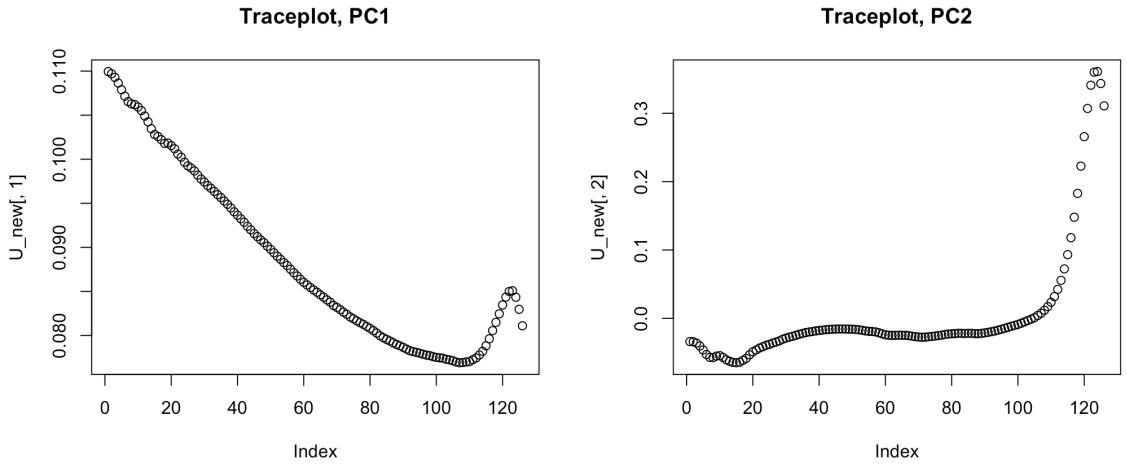


We can see that the majority of the variance is covered by the first component, with PC1 covering 93.332 % and PC2 covering 6.263 %.



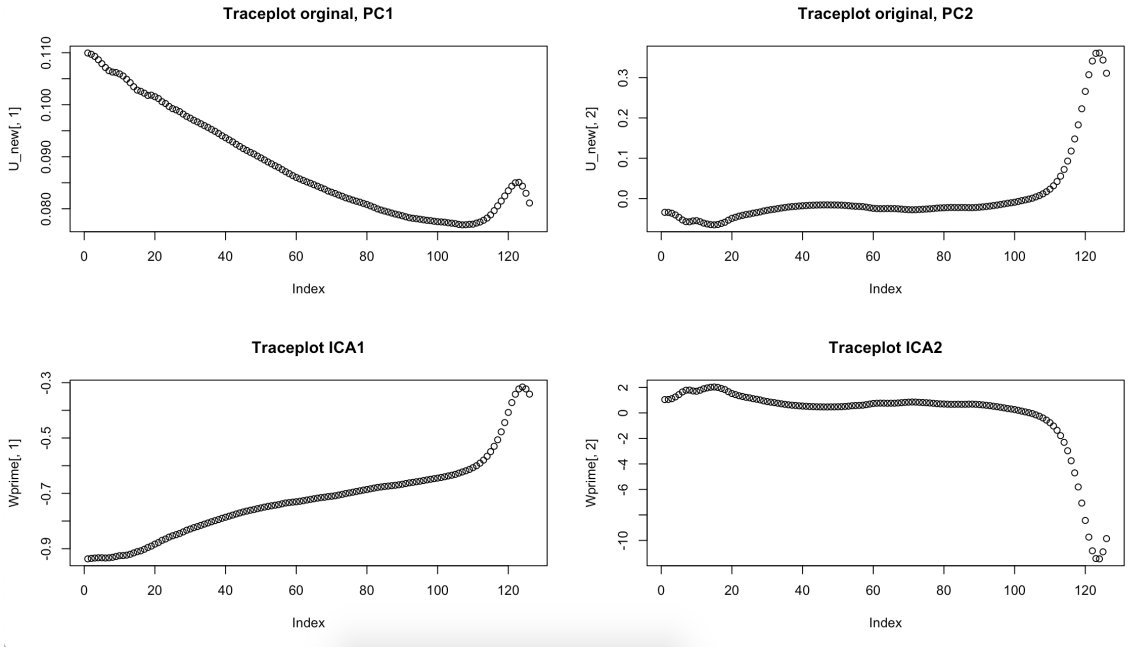
With the plot of the scores we can see which datapoints that are similar to each other. Here, it is also shown as in the first figure that most data are in one single cluster. In the traceplots we can see that there are very few unusual diesel fuels (to the right), since there are single datapoints that don't create a cluster.

2. Perform Independent Component Analysis with the number of components selected in step 1. Is there a PC that is explained by only a few features?



With traceplots we can see the correlation. PC1 can be described by almost all features, specially by the early ones. PC2 can be well described by the last features.

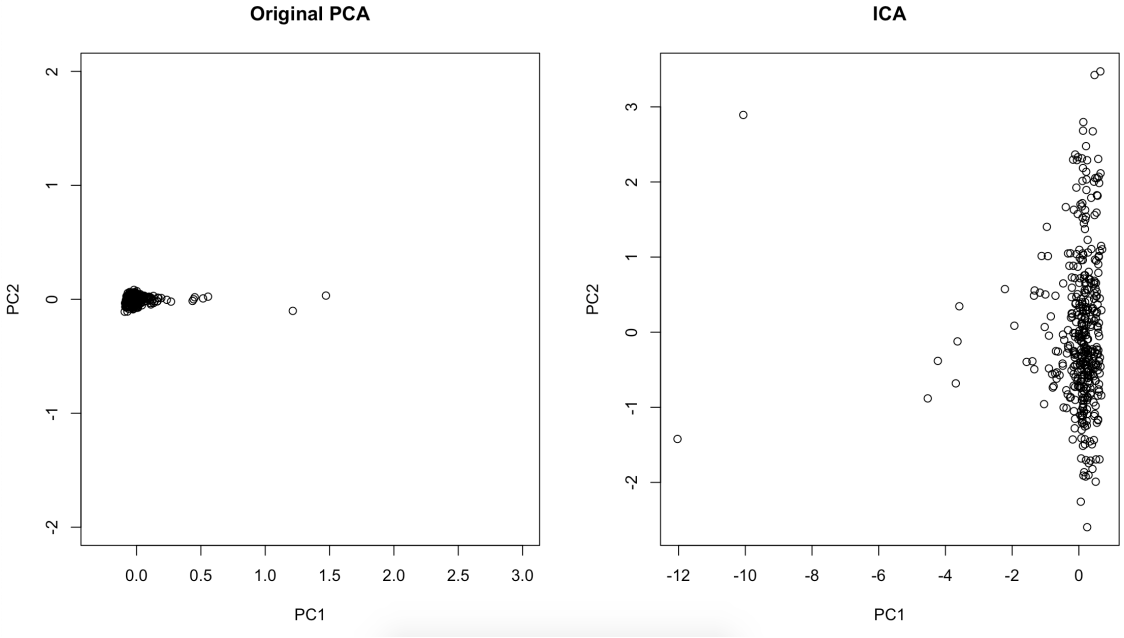
3. Perform ICA with the chosen components selected from step 1. Make a plot of the scores of the first two latent features and compare it with the first score plot.



When using the PCA method, vectors with maximum variance and dependence is found meanwhile ICA finds vectors that are independent, with no correlation when making a

transformation to a new feature space. This results in mirroring when a rotation is made, as we can see in the traceplots. W prime explains how much each component are described by the original features, as we can see in the traceplots.

With traceplot we can see correlation from features, where for the first variable are dependent by the early indexes, both when using the ICA and PCA methods (except it's positive or negative beacuse of the "reflection"). Same conclusion can be made when comparing the traceplots for the second variable for ICA and PCA, where it's mostly dependent by the higher indexes.



The score for ICA is similar to the score for PCA, except it is scaled and "mirrored".

APPENDIX CODE ----------

```
setwd("~/Studier+Studentliv/HT19/TDDE01/Labb2")
crabs=read.csv("australian-crabs.csv")

#1
plot(crabs[["CL"]], crabs[["RW"]], main="Black=Female, Red=Male First Correlation",
     xlab="CL", ylab="RW", col=crabs$sex)

#2 LDA ------------------------------------------------------
#install.packages("MASS")
library("MASS")

misscon_rate = function (conf_mat, fit_mat) {
  n=length(fit_mat[,1])
  miss_rate=(1-sum(diag(conf_mat))/n)
  print(miss_rate)
}

fitLda = lda(crabs$sex ~ CL + RW, data = crabs)
predict = predict(fitLda, newdata=crabs)

confusionmatrix<- table(crabs$sex, predict$class)

#crabs$sex - real values, predicted$class gives for each value i it was predicted
male or female
misscon_rate(confusionmatrix, crabs)
#missclassification rate = 0.035

plot(crabs[["CL"]], crabs[["RW"]], main="Black=Female, Red=Male Predicted",
     xlab="CL", ylab="RW", col=predict$class)

#-----3 set prior(male) =0.9, prior(female)=0.1 - set in LDA -

new_prior_lda = lda(crabs$sex ~ CL + RW, data = crabs, prior = c(0.1,0.9))
#prior är kopplat till sex här
#level = möjligheter som är female och male som vi kan klassa till

predict_new_prior = predict(new_prior_lda, newdata=crabs)
confusionmatrix_new_prior<- table(crabs$sex, predict_new_prior$class)
misscon_rate(confusionmatrix_new_prior, crabs) # missclassification rate = 0.08

plot(crabs[["CL"]], crabs[["RW"]], main="Black=Female, Red=Male, New prior",
     xlab="CL", ylab="RW", col=predict_new_prior$class)
# We can se that there are more males in this plot
#Coefficients of linear discriminants:
#  LD1
#CL  0.5765241
#RW -1.6823062

# 4 ------logistic regression instead of LDA------------------

fit_logistic = glm(sex ~ CL + RW, data = crabs, family='binomial')

predict_logistic = predict(fit_logistic, newdata=crabs, type='response')
confusionmatrix_logistic<- table(crabs$sex, predict_logistic > 0.5)
misscon_rate(confusionmatrix_logistic, crabs) # missclassification rate = 0.03

plot(x=crabs[["CL"]], y=crabs[["RW"]], main="Black=Female, Red=Male, Logistic
Regression",
     xlab="CL", ylab="RW", col=ifelse (predict_logistic >0.5, "red", "black" ),
ylim=c(0,20),xlim=c(0,50))

#Since we have very "kind" data, it is very easy to make create a good fit for the
model, giving us good prediction for each datapoint.
#This gives us very low misclassification rate, but this can also be explained by
that we use the same data when doing the prediction as when we do the fit of the
model
```

```
#Equation of decision boundary (since ln(0.5/0.5)=ln(1)=0 --> )

beta_0 = fit_logistic$coefficients[1]
beta_1 = fit_logistic$coefficients[2]
beta_2 = fit_logistic$coefficients[3]

CLIntercept <- - beta_0 / beta_2
slope <- - beta_1 / beta_2

par(new=TRUE)
curve(slope*x + CLIntercept,
                    from=0, to=50, add=TRUE, col="green")




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

setwd("~/Studier+Studentliv/HT19/TDDE01/Labb2")
credit=read.csv2("creditscoring.csv")
RNGversion('3.5.1')

misscon_rate = function (conf_mat, fit_mat) {
  n=length(fit_mat[,1])
  miss_rate=(1-sum(diag(conf_mat))/n)
  print(miss_rate)
}

#1 Splitta data
n=dim(credit)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=credit[id,]

id1=setdiff(1:n, id)
set.seed(12345)

id2=sample(id1, floor(n*0.25))
valid=credit[id2,]

id3=setdiff(id1,id2)
test=credit[id3,]

#install.packages("tree")
library("tree")

#2 ------------------------------------------------
par(mfrow = c(1,1))
#DEVIANCE
fit_tree_dev = tree(good_bad~., data=train, split="deviance")

predict_train_dev = predict(fit_tree_dev, newdata=train, type="class")
predict_test_dev = predict(fit_tree_dev, newdata=test, type="class")

#misclassification rates with deviance
CM_dev_train<- table(train$good_bad, predict_train_dev)
misscon_rate(CM_dev_train, train) #misclassification rate = 0.212

CM_dev_test<- table(test$good_bad, predict_test_dev)
misscon_rate(CM_dev_test, test) #misclassification rate = 0.268

#GINI

fit_tree_gini = tree(good_bad~., data=train, split="gini")
```

```
predict_train_gini = predict(fit_tree_gini, newdata=train, type="class")
predict_test_gini = predict(fit_tree_gini, newdata=test, type="class")

#misclassification rates with deviance
CM_gini_train<- table(train$good_bad, predict_train_gini)
misscon_rate(CM_gini_train, train) #misclassification rate = 0.242

CM_gini_test<- table(test$good_bad, predict_test_gini)
misscon_rate(CM_gini_test, test) #misclassification rate = 0.372

#We get better values with the split from deviace, and will continue with that one.


#3---- Use training av validation sets in order to choose the optimal tree depth. -
---------

#We take the fitted tre with deviance from #2 !

#create vectors for differents depths to try (?)
trainScore=rep(0,12)
testScore=rep(0,12)

for (i in 2:12) {
  prunedTree=prune.tree(fit_tree_dev, best=i)
  predTree = predict(prunedTree, newdata=valid, type="tree") #We don't need to do
this for the trained data
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(predTree)
}

plot(2:12, trainScore[2:12], type = "b", col="red", ylim = c(200,700),
ylab="deviance", xlab="amount of leaf nodes")
points(2:12, testScore[2:12], type = "b", col="blue")

prunedTree_4=prune.tree(fit_tree_dev, best=4)
predTree_4 = predict(prunedTree_4, newdata=valid, type="class")

CM_tree_dev <- table(valid$good_bad, predTree_4)
misscon_rate(CM_tree_dev, valid) #misclassification rate=0.264


plot(prunedTree_4)
text(prunedTree_4, pretty=0)

CM_tree_dev


#-----4-----------------NAIVES----------------------------

#install.packages("MASS")
#install.packages("e1071")
library("MASS")
library("e1071")

#Predict Values for Naives
fit_NB = naiveBayes(good_bad~., data=train)

pred_NB_train = predict(fit_NB, newdata=train)
pred_NB_test = predict(fit_NB, newdata=test)

#Misclassification rates
CM_NB_train<- table(train$good_bad, pred_NB_train)
misscon_rate(CM_NB_train, train) #misclassification rate = 0.3

CM_NB_test<- table(test$good_bad, pred_NB_test)
misscon_rate(CM_NB_test, test) #misclassification rate = 0.316 #much better - why a
better model??
```

```
#---------5--------------------
#Loopa för varje pi - för ett pi - gör classification från FIT_model (först tree
sen NB) med sannoliketer för good - större än pi? --> Y=1 (good)!!!

#Create pi vector

get_FPR_TPR = function (pred_fit){

  pi=seq(0.05, 0.95, by=0.05)
  TPR = c()
  FPR = c()

  #pred_opti_tree = predict(prunedTree_4, newdata=test, type="vector")

  for (pi_value in pi) {
    #temporär classify vektor - ny för varje pi!
    own_classify=c()
    for (value in pred_fit[,2]){  #Only take the good values!
      if (pi_value > value){
        own_classify=c(own_classify, "bad")
      } else {
        own_classify=c(own_classify, "good")
      }
    }

    #for each pi - get the confusion matrix!
    CM = table(valid$good_bad, own_classify)

    #If only a vector - fill the other column. Get the right format
    if (ncol(CM) == 1) {
      if(is.element("bad", colnames(CM))) {
        CM = cbind("good" = c(0,0), CM)
      } else {
        CM = cbind(CM, "bad" = c(0,0))
      } }

    # Computing TPR and FPR from the classification matrix
    TPR_value = CM[2,2]/sum(CM[2,])
    FPR_value = CM[1,2]/sum(CM[1,])
    # Adding the value to the TPR and FPR vectors
    TPR = c(TPR, TPR_value)
    FPR = c(FPR, FPR_value)
  }
  return (cbind(TPR, FPR))
}

#tree
pred_opti_tree = predict(prunedTree_4, newdata=valid, type="vector")
ROC_tree = get_FPR_TPR(pred_opti_tree)

#NB
pred_opti_NB = predict(fit_tree_dev, newdata=valid, type="vector")
ROC_NB = get_FPR_TPR(pred_opti_NB)

#sort the tree ROC values for correct plotting....
ROC_TPR_tree=sort(ROC_tree[,1])
ROC_FPR_tree=sort(ROC_tree[,2])
ROC_tree_sorted = cbind(ROC_TPR_tree, ROC_FPR_tree)

#Plot for Tree ROC and NB ROC
plot(ROC_tree_sorted[,2], ROC_tree_sorted[,1], col="green",type="l", xlab="FPR",
ylab="TPR", ylim=(0:1), xlim=(0:1))
par(new=TRUE)
plot(ROC_NB[,2], ROC_NB[,1], col="blue",type="l", xlab="FPR", ylab="TPR",
ylim=(0:1), xlim=(0:1))

#For most thresholds, the tree classification gives better results comapred to when
doing with NB, except for some thresholds.
```

```
# ---- 6 ------ instead of using pi, we use 10/11 as threshold -- NAIVE BAYES----
fit_NB = naiveBayes(good_bad~., data=train)

pred_NB_test = predict(fit_NB, newdata=test, type="raw")
pred_NB_train = predict(fit_NB, newdata=train, type="raw")

#Se föreläsningsanteckningar
CM_test = table(test$good_bad, pred_NB_test[,2]/pred_NB_test[,1] > 10/1)
misscon_rate(CM_test, test) #=0.508

#FALSE TRUE
#bad     71    5
#good   122   52

CM_train = table(train$good_bad, pred_NB_train[,2]/pred_NB_train[,1] > 10/1)
misscon_rate(CM_train, train) #=0.546

#FALSE TRUE
#bad    137   10
#good   263   90









#-------------- Assignment 4 ---------------------------

setwd("~/Studier+Studentliv/HT19/TDDE01/Labb2")
data_frame=read.csv2("NIRSpectra.csv")
RGNversion('3.5.1')

# ----1 -----Principal Component Analysis - PCA
source("https://bioconductor.org/biocLite.R")
biocLite("pcaMethods")

data_frame$Viscosity=c()
PCA=prcomp(data_frame)
lambda=PCA$sdev^2

#eigenvalues - bygger upp rummet
lambda

par(mfrow = c(1,1))
#proportion of variation
prop_vari = sprintf("%2.3f",lambda/sum(lambda)*100) #Visar varians för varje PC.
(typ egenvärde i rummet)
screeplot(res)

#Scores
#See how the variables cluster??? - Shows which variables are similar
plot(PCA$x[,1], PCA$x[,2], ylim=c(-0.3,0.6)) #datapunkterna i nya
koordinatsystemet,

#----- 2 TRACE PLOTS-------
#första står för störst variation - tar bort de andra axlarna och använda sig utav
färre dimensioner
U_new=PCA$rotation
head(U)

#loading - shows which variables are correlated!
par(mfrow = c(1,2))
```

```
plot(U_new[,1], main="Traceplot, PC1") #alla varaibler kring 01 - index= feauture
(högt värde = PC1 beskriver den features till hög grad)
plot(U_new[,2],main="Traceplot, PC2")

#------ 3 ---------
library("fastICA")
set.seed(12345)

#ICA antar att variablerna är inte beroende - kan hitta andra band
result_ICA = fastICA(data_frame, 2, fun = "logcosh", alpha = 1, row.norm = FALSE)
Wprime = result_ICA$K%*%result_ICA$W  #matrix multiplication

# Traceplots
par(mfrow = c(2,2))

plot(U_new[,1], main="Traceplot orginal, PC1")
plot(U_new[,2],main="Traceplot original, PC2")

plot(Wprime[,1], main="Traceplot ICA1")
plot(Wprime[,2], main="Traceplot ICA2")

# Scores
par(mfrow = c(1,2))
res=prcomp(data_frame)
plot(res$x[,1], res$x[,2], ylim=c(-2,2), xlim=c(-0.3,3), xlab="PC1", ylab="PC2",
main="Original PCA")
plot(result_ICA$S[,1], result_ICA$S[,2], xlab="PC1", ylab="PC2", main="ICA")
```