Report_Lab2

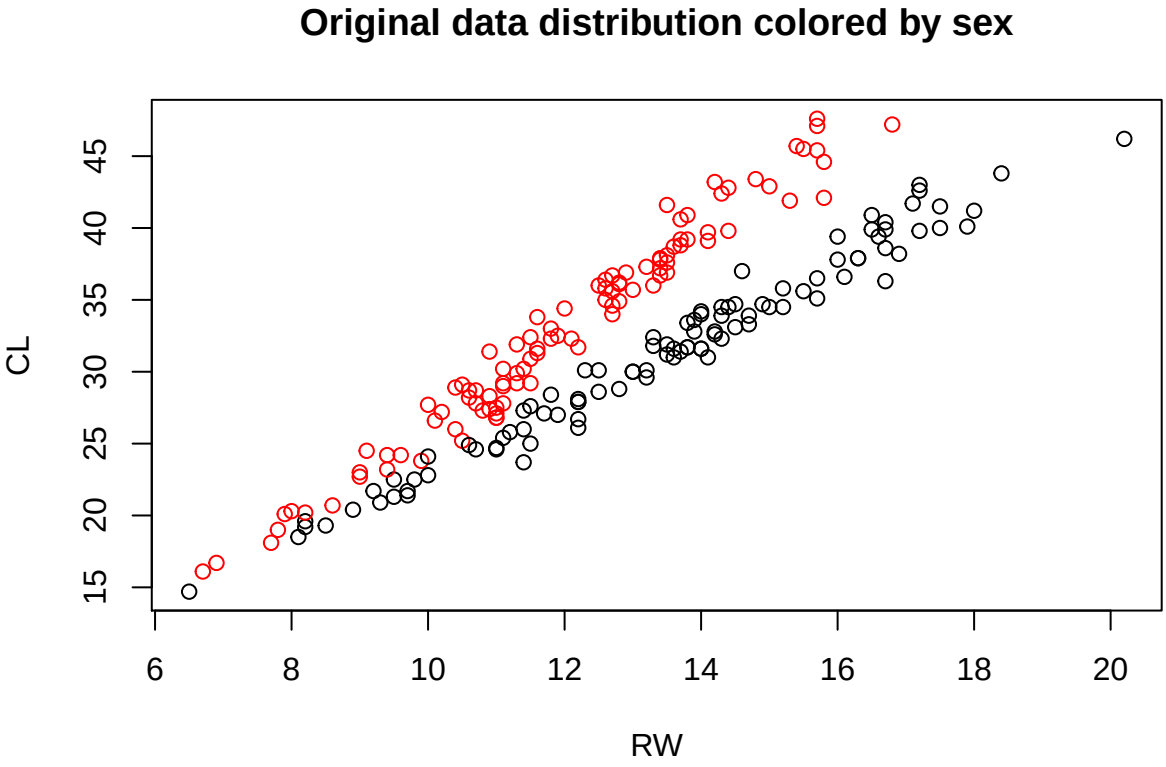
Karol Wojtulewicz

Assignment 1

In this assignment we used the dataset "australian-crabs.csv".

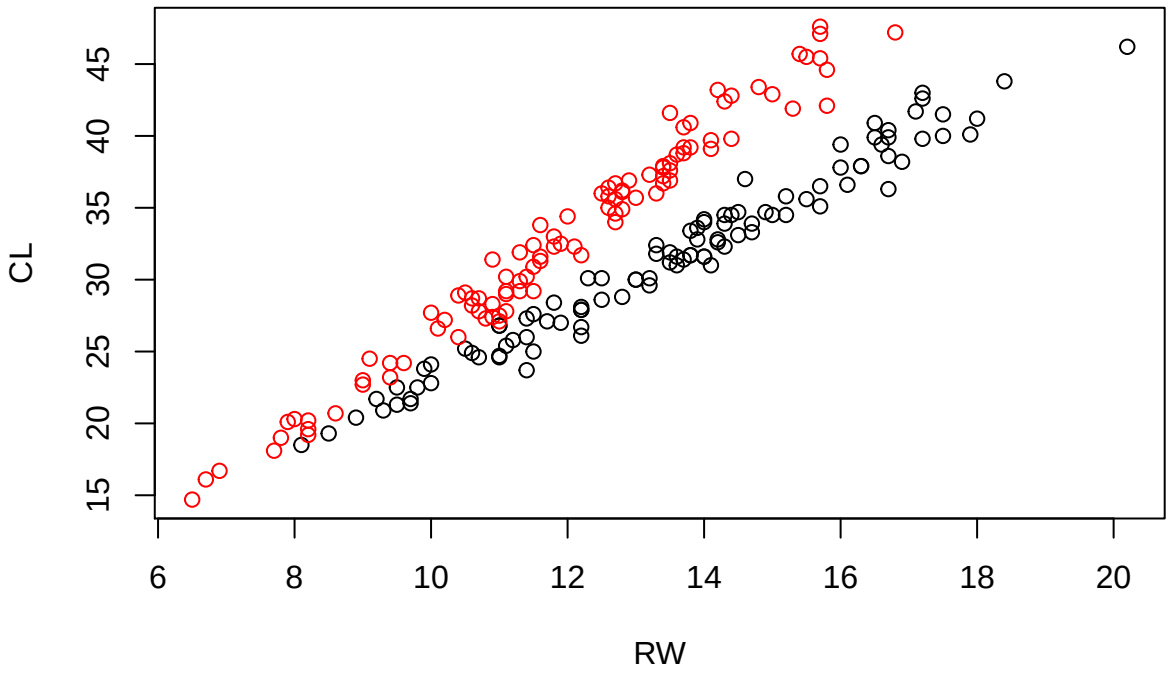
Assignment 1.1

Here we were supposed to plot the data from the dataset, RW against the CL and colour it by sex. The dataset looks to be suited for the discriminant analysis, because it is mainly divided in two chunks of data with some overlapping points near the origin of the graph.



Assignment 1.2

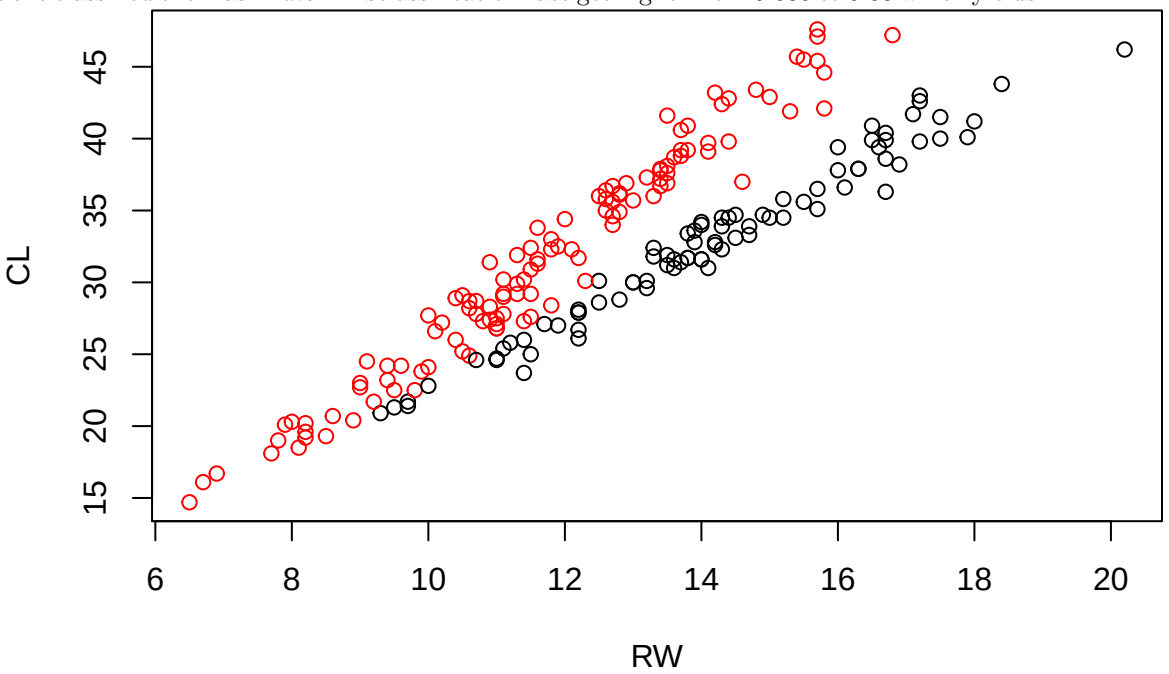
In this sub assignment we were supposed to (as in the previous assignment) plot the data from the plot and in addition use lda to classify the points, and use the prediction to colour them. Here we can see slight differences in how the LDA classified the points comparing to the original classes. As mentioned before, points near the origin of the graph cause some misclassification, but as we can see it is not significant as we can see in the print out underneath the graph.




```
## [1] "Missclassification:  0.035"
```

Assignment 1.3

In this sub assignment we were supposed to repeat the steps from the previous assignment, but with the prior of $p(Male) = 0.9, p(Female) = 0.1$. As wee can see below, some of the black dots form the previous graph turned red, indicating that the classified them as $Male$. Misclassification rate get higher from 0.035 to 0.08 which yields



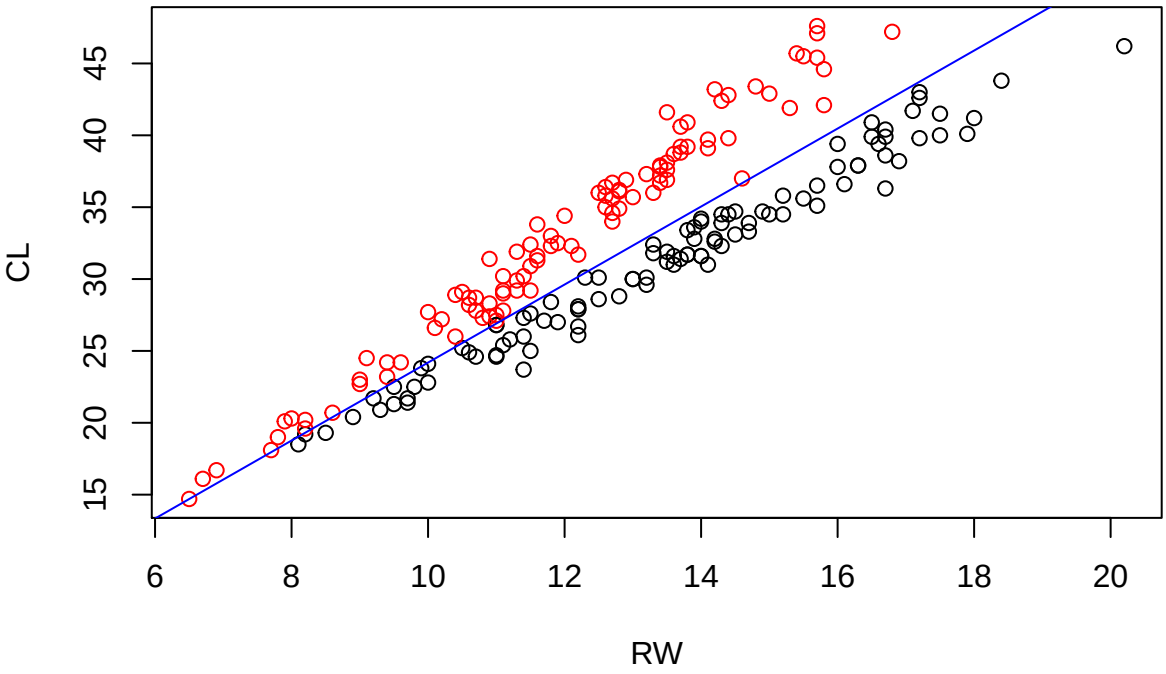
a worsen classifier.

```
## [1] 0.08
```

Assignment 1.4

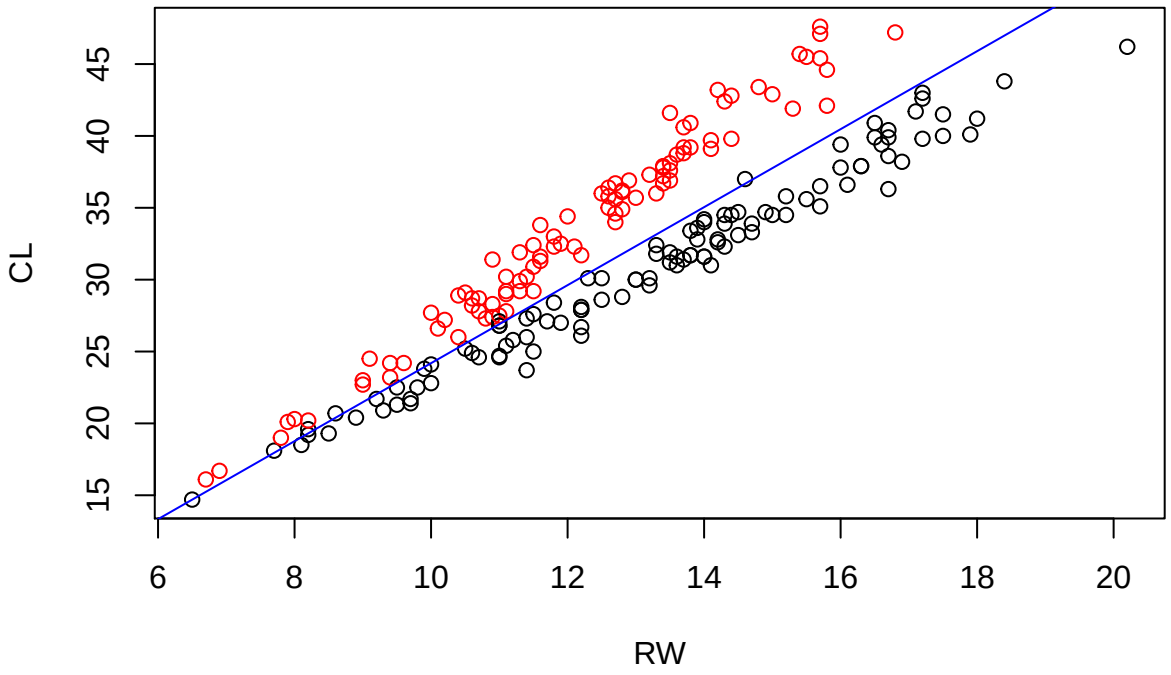
Here we shall repeat the steps 1.2 and 1.3 but with logistic regression. The decision boundary in this case is $logit(t) = \ln(\frac{t}{1-t})$ where t is our threshold (0.5 and 0.9). We can see in the graphs, that the story is not quite similar as it was with LDA, with slight differences (like the alone dot in the middle of the graph) and the same classification rate. We can see in the lower graph, how the points on the decision line turn black which is the result of our requirement $p(Male) = 0.9, p(Female) = 0.1$.

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




```
## [1] 0.035
```

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




```
## [1] 0.035
```

Assignment 2

In this assignment we used the "creditscoring.xls" dataset and created and evaluated Naive Bayes and Decision Tree models.

Assignment 2.2

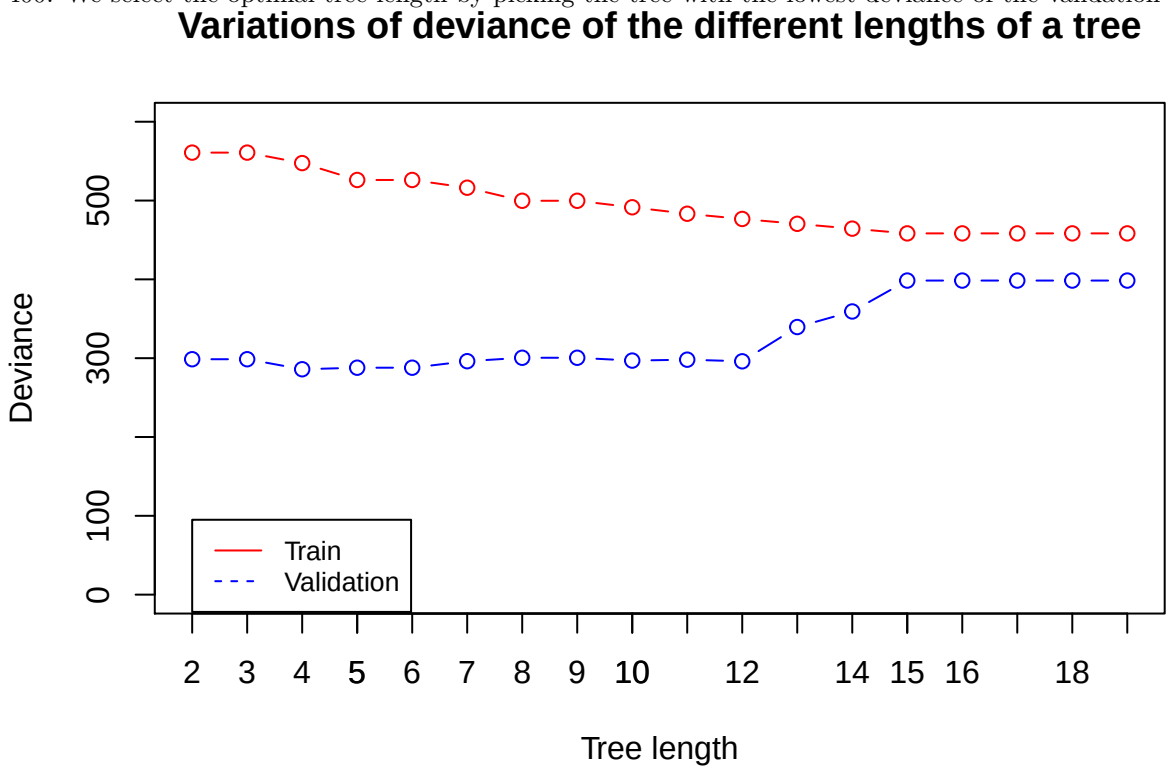
In this sub assignment the task was to fit the decision tree to the previously created training subset of the provided dataset. We used two different measures of impurity: Deviance and Gini index. Misclassification rates provided below show, that the model created with deviance impurity is better, showing lower rate than the gini index for both training and testing datasets. The significant difference in misclassification with gini index between training (Msr=0.238) and testing (Msr=0.372) datasets show the overfitting of the model, where as for deviance model (train = 0.212 and test = 0.268), the results show the better performance. Thus the model created with the deviance impurity has been selected for further investigation.

```
## [1] "Misclassification deviance train:  0.212"
## [1] "Misclassification deviance test:  0.268"
## [1] "Misclassification gini train:  0.238"
## [1] "Misclassification gini test:  0.372"
```

Assignment 2.3

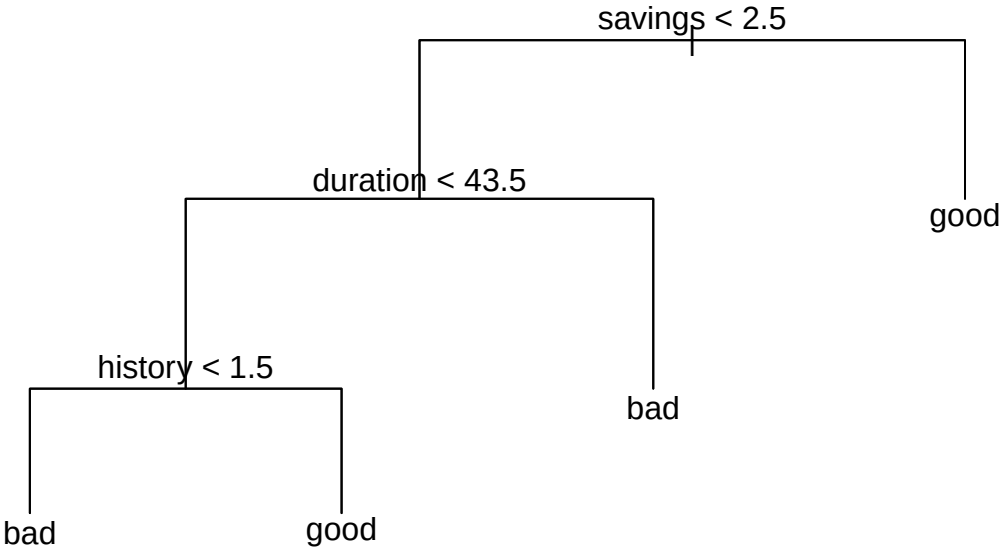
In this sub assignment the task was to find the optimal tree depth of the previously selected model and provide a graph of how the deviance varies with the changeing tree length. In the graph shown below we can see how the deviance varies with the changing tree length. Lower tree lengths report high deviance for the training dataset and low for the validation dataset. Both datasets' deviance seem to converge towards the value of

400. We select the optimal tree length by picking the tree with the lowest deviance of the validation dataset.




```
## [1] "Best length: 4"
```

The selected tree length is 4 and the tree is printed below. It consists of three variables: savings, duration and history. The tree is printed below and it shows a reasonable decision process, where we can see how a person managed their loans depending on savings (if it does not have enough savings algorithm iterates to the duration node). The tree seem to reason logically and the thought process can be easily understood. When we measure the misclassification rate of the selected tree length, we can see that it is lower, than the original, unpruned tree and the Msr value is 0.256 comparing to 0.268 of the original tree.




```
## [1] 0.256
```

Assignment 2.4

In this subassignment the task was to train and evaluate a Naive Bayes model using a confusion matrix with training and test data as well as the classification rates. In the printouts seen below we can see that the performance seems to be worse than the one of the selected decision tree, yielding the Msr of 0.3 for the train dataset and 0.316 for the test. This result shows, that the choise of the selected decision tree model could be a better one.

```
##
##                Bad Good
##   Actual Bad    95   52
##   Actual Good   98  255
```

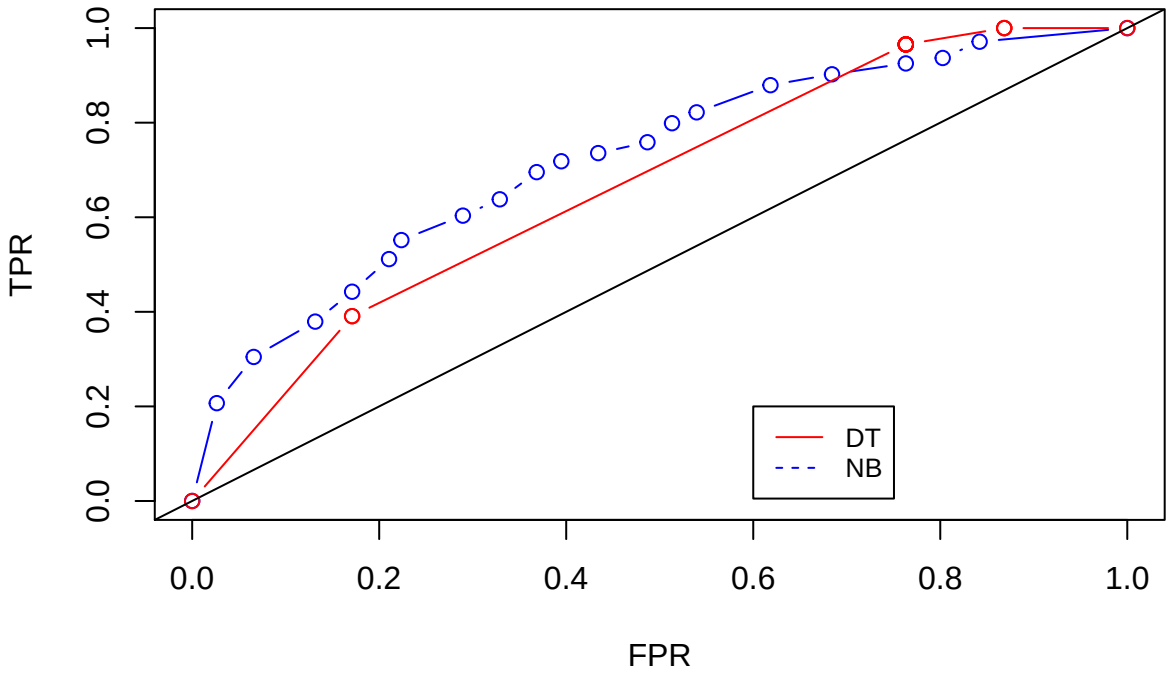
```
##
##                Bad Good
##   Actual Bad    46   30
##   Actual Good   49  125
```

```
## [1] "Misclassification naive bayes train:  0.3"
```

```
## [1] "Misclassification naive bayes test:  0.316"
```

Assignment 2.5

In this subassignment the task was to evaluate the performance of both the Naive Bayes and the selected decision tree models using a ROC (Receiver operating characteristic) curve with a threshold varying between 0.05 and 0.95 with 0.05 step. The plot below shows how the two models perform. We can deduce a performance of a model by looking at the integral of the curve (area underneath the curve), bigger is better. In the graph we can see, that the Naive Bayes model performed better than the selected tree one. It is worth mentioning, that the decision tree model did not plot many points on the graph (corresponding to TPR and FPR) due to the lack of sufficient confusion matrix. Above a certain treshold (around 0.7) of probability the model predicted just false values, leading to a uniform vector of 0. This means, that the model did not provide a sufficient enough certainty, making it a bad choice for classification in comparison to the Naive Bayes model.



Assignment 2.6

In this assignment the task was to repeat the classification as it was in the step 4, but with the following loss matrix: $L_{Observed} = \begin{pmatrix} 0 & 1 \\ 10 & 0 \end{pmatrix}$ (first row is actual good and second actual bad). The Msr results below show worsen performance comparing to the first Naive Bayes model with lower values for both train (0.546) and test (0.508) datasets. The impact of the loss matrix is more pronounced in the confusion matrix, where the field false-positive has significantly less values than the field false-negative.

```
##
##                Bad Good
##   Actual Bad   137   10
##   Actual Good 263   90
```

```
##
##                Bad Good
##   Actual Bad    71    5
##   Actual Good 122   52
```

```
## [1] "Missclassification train 0.546"
```

```
## [1] "Missclassification test 0.508"
```

If we transpose the loss matrix to $L_{Observed} = \begin{pmatrix} 0 & 10 \\ 1 & 0 \end{pmatrix}$ where we "punish" more the false positive values, both the confusion matrix and the Msr score is better than the one observed above, which means a better model.

```
##
##                Bad Good
##   Actual Bad    27  120
##   Actual Good   17  336
```

```
##
##                Bad Good
##   Actual Bad    14   62
##   Actual Good   10  164
```

```
## [1] "Missclassification train 0.274"
```

```
## [1] "Missclassification train 0.288"
```

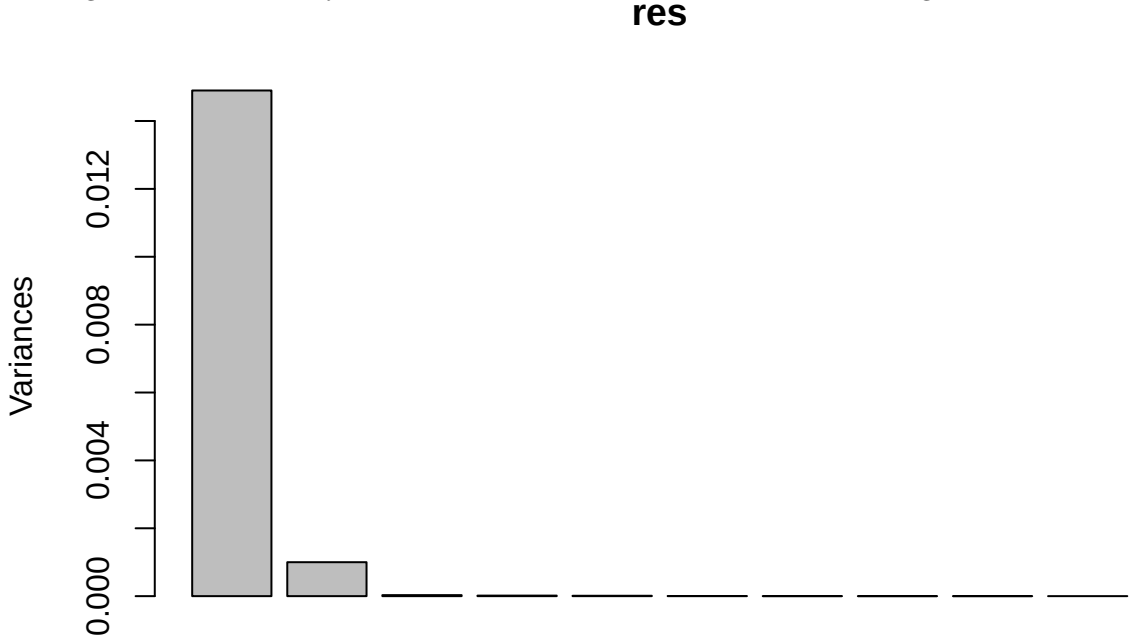
Assignment 4

In this assignment we use a dataset "NIRspectra.csv" that contains "near-infrared spectra and viscosity levels for a collection of diesel fuels" as said in the laboration file. The task is to use PCA and ICA to reduce dimentionality of the dataset.

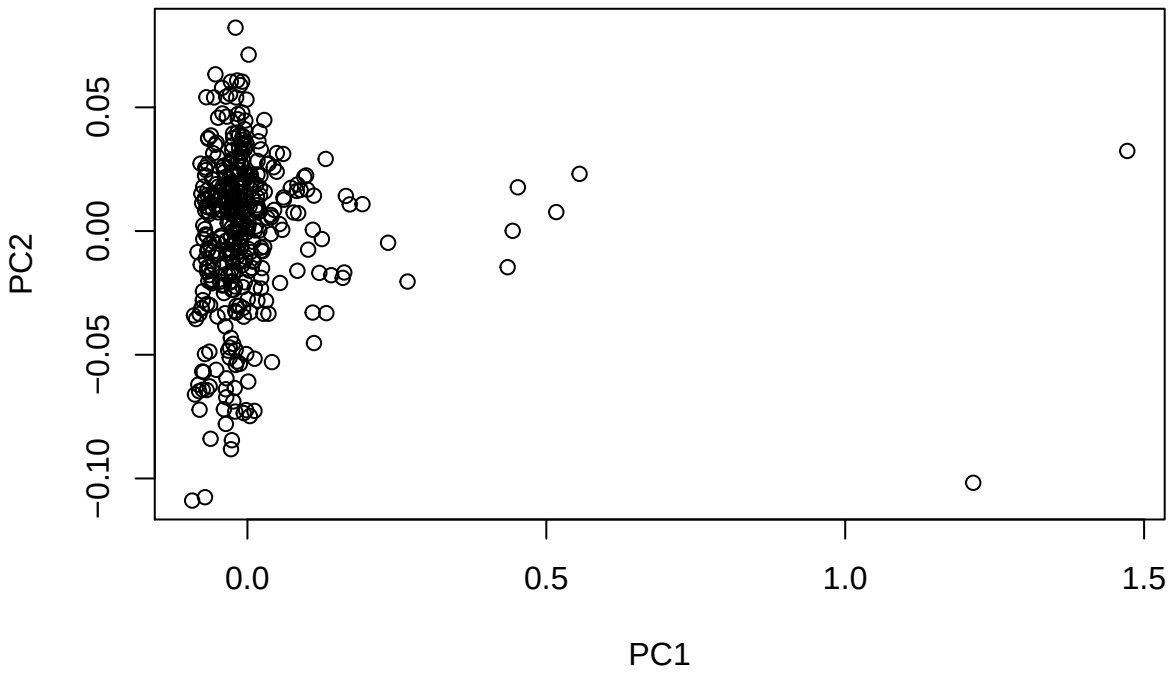
Assignment 4.1

In this assignment the task is to create a plot showing PC values, axis in which the data varies the most in the dataset. The first PC, PC1 gives the axis of the biggest variation of the data, PC2 the second most variation and so on. The values below show how much variation catch 6 first PCs (there are 126 PCs in total, but just the first 6 are printed for convenience). We can see, that PC1 catches over 93% of the variation

and together with PC2 they catch over 99% of the variation. Below the histogram of the PCs is provided.

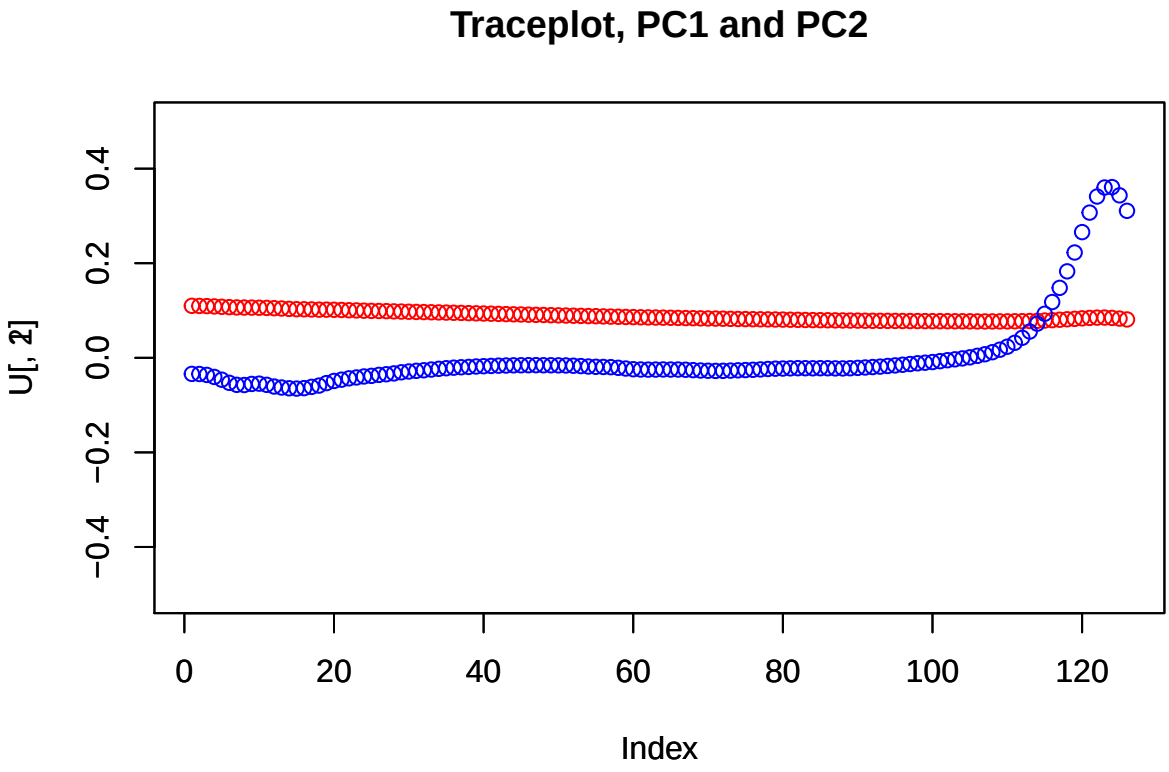


The plot privided below shows the datapoints in the PC1, PC2 coordinate system. The points that are most further out (around 1-1.5) are the unusual diesel fuels. High majority of the fules are very similar.



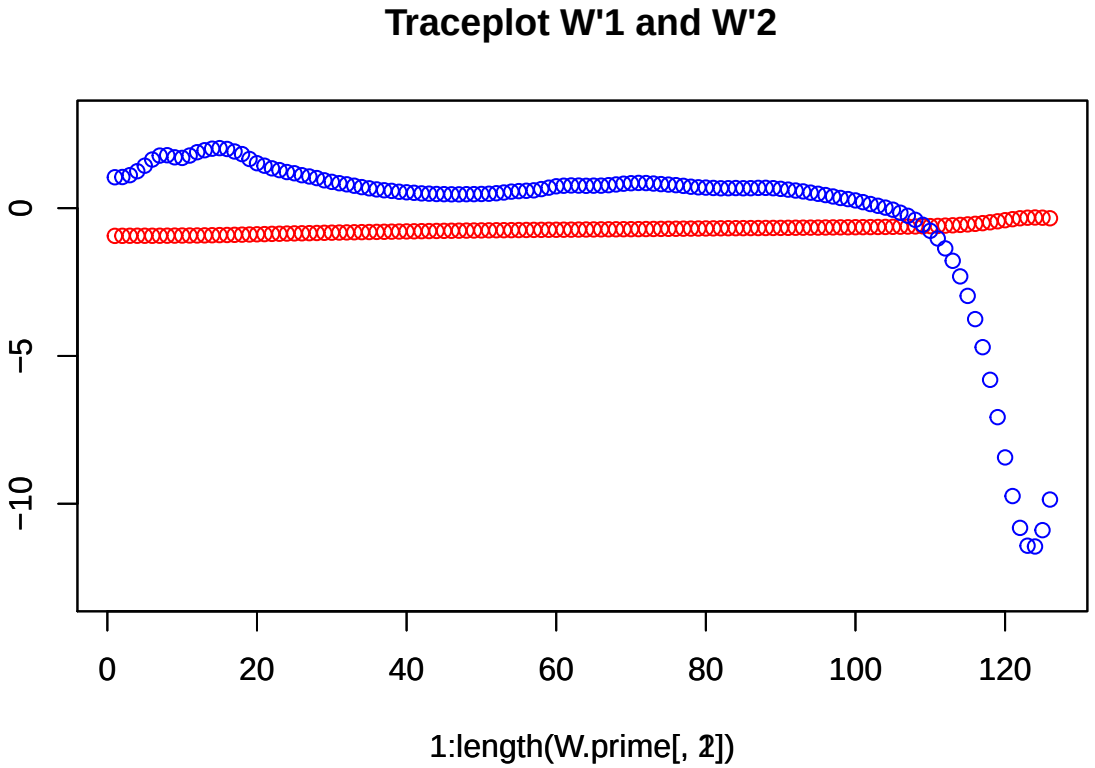
Assignment 4.2

In this assignment the task was to create the trace plots of the principal components, that catch 99% of the variations in the data. In the graphs below we can see the plots. Both of the plots show a unique behaviour of the data. In the PC2 the data up to 100:th feature seem to behave linear, but it has a spike around 120. None of these components can be described with just a few of the original features.



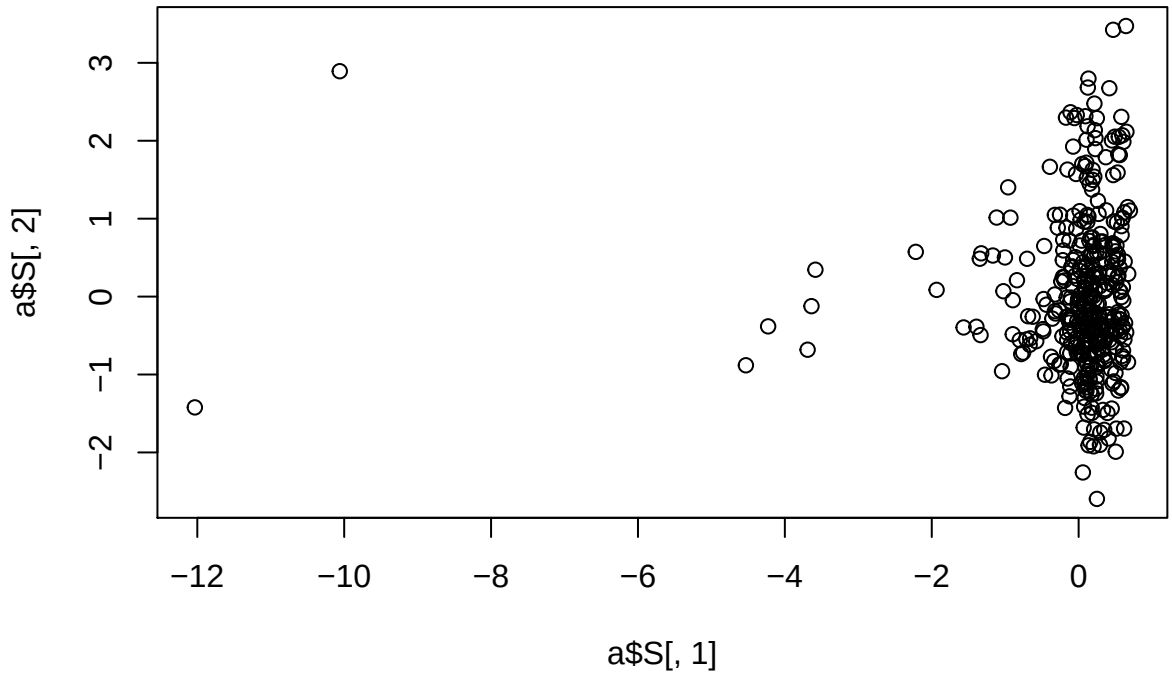
###

Assignment 4.3 In this assignment the task was to perform fastICA, traceplot W' = K*W and compare it to the plots from step 2. We also needed to create a scoreplot. As we can see below, the traceplots are very similar to the ones found in the step 2 with two main differences: the plots have different scaling and are mirrorred.



Below can we see scoreplot from the first latent features we got from fastICA. It is very similar to the step 1

but mirrored and with different scaling.



Code

Assignment 1

```
data = read.csv(file="/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab\ 2/australian-crabs.cs
set.seed(12345)
# Assignment 1.1
plot(data$RW, data$CL, col= data$sex, xlab = "RW", ylab = "CL")

# Assignment 1.2
#LDA - Linear Discriminant Analysis
data = read.csv(file="/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab\ 2/australian-crabs.cs
set.seed(12345)
library(MASS)
model.lda = lda(sex~RW+CL, data= data)
predictor.dataframe = data.frame(RW = data$RW, CL = data$CL)
colnames(predictor.dataframe) = c("RW", "CL")
predict.lda = predict(model.lda, predictor.dataframe)
plot(data$RW,data$CL, col= predict.lda$class, xlab = "RW", ylab = "CL")
misClasificError = mean(predict.lda$class != data$sex)
print(misClasificError)
#Really good fit, misclassification error: 0.035. You can barelly see differences.

# Assignment 1.3
#LDA - Linear Discriminant Analysis
data = read.csv(file="/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab\ 2/australian-crabs.cs
set.seed(12345)
library(MASS)
model.lda = lda(sex~RW+CL, data= data, prior = c(0.9,0.1))
predictor.dataframe = data.frame(RW = data$RW, CL = data$CL)
```

```r
colnames(predictor.dataframe) = c("RW", "CL")
predict.lda = predict(model.lda, predictor.dataframe)
plot(data$RW,data$CL, col= predict.lda$class, xlab = "RW", ylab = "CL")
misClasificError = mean(predict.lda$class != data$sex)
print(misClasificError)
#Really good fit, misclassification error: 0.08. You can barelly see differences.

# Assignment 1.4a
#LDA - Linear Discriminant Analysis
data = read.csv(file="/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab\ 2/australian-crabs.cs
set.seed(12345)

model.log.reg = glm(sex~RW+CL,family=binomial(link='logit'), data= data)
predictor.dataframe = data.frame(RW = data$RW, CL = data$CL)
colnames(predictor.dataframe) = c("RW", "CL")
predict.log.reg = predict(model.log.reg, predictor.dataframe, type = "response" )
slope <- coef(model.log.reg)[2]/(-coef(model.log.reg)[3])
intercept <- coef(model.log.reg)[1]/(-coef(model.log.reg)[3])

predict.log.reg = ifelse(predict.log.reg > 0.5,"Male","Female") #Assignment 1.2
dataframe.sex = data.frame(sex = predict.log.reg)
plot(data$RW,data$CL, col= dataframe.sex$sex, xlab = "RW", ylab = "CL")
abline(intercept, slope, col = "blue")

misClasificError = mean(dataframe.sex$sex != data$sex)
print(misClasificError)
#Really good fit, misclassification error: 0.035. You can barelly see differences.
#Same result as in 1.2

# Assignment 1.4b
#LDA - Linear Discriminant Analysis
data = read.csv(file="/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab\ 2/australian-crabs.cs
set.seed(12345)

model.log.reg = glm(sex~RW+CL,family=binomial(link='logit'), data= data)
slope <- coef(model.log.reg)[2]/(-coef(model.log.reg)[3])
intercept <- coef(model.log.reg)[1]/(-coef(model.log.reg)[3])
predictor.dataframe = data.frame(RW = data$RW, CL = data$CL)
colnames(predictor.dataframe) = c("RW", "CL")
predict.log.reg = predict(model.log.reg, predictor.dataframe, type = "response" )

predict.log.reg = ifelse(predict.log.reg > 0.9,"Male","Female") #Assignment 1.2
dataframe.sex = data.frame(sex = predict.log.reg)
plot(data$RW,data$CL, col= dataframe.sex$sex, xlab = "RW", ylab = "CL")
abline(intercept, slope, col = "blue")

misClasificError = mean(dataframe.sex$sex != data$sex)
print(misClasificError)
#Really good fit, misclassification error: 0.035. You can barelly see differences.
#Same result as in 1.2
```

Assignment 2

```r
data = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab\ 2/credits

#Assignment 2.1
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]
id3=setdiff(id1,id2)
test=data[id3,]

#Assignment 2.2
train = as.data.frame(train)
library(tree)
label = ifelse(train$good_bad == "good", 2,1)
#train$good_bad = label
#test$good_bad = ifelse(test$good_bad == "good", 2,1)
#valid$good_bad = ifelse(valid$good_bad == "good", 2,1)
n = dim(train)[1]
fit.deviance = tree(as.factor(good_bad)~. , train, split = c("deviance")) #as.factor used from this pos
fit.gini = tree(as.factor(good_bad)~. , train, split = c("gini")) #as.factor used from this post:
#https://stackoverflow.com/questions/31843212/r-predict-func-type-class-error

#plot
#plot(fit.deviance)
#text(fit.deviance, pretty=0)
plot(fit.gini)
text(fit.gini, pretty=0)
fit1
#summary(fit1)

#predic
predict.fit.deviance.train=predict(fit.deviance, newdata=train, type="class")
predict.fit.deviance.test=predict(fit.deviance, newdata=test, type="class")
predict.fit.gini.train=predict(fit.gini, newdata=train, type="class")
predict.fit.gini.test=predict(fit.gini, newdata=test, type="class")

#misclass error
misClasificError.deviance.train = mean(predict.fit.deviance.train != train$good_bad)
misClasificError.deviance.test = mean(predict.fit.deviance.test != test$good_bad)
misClasificError.gini.train = mean(predict.fit.gini.train != train$good_bad)
misClasificError.gini.test = mean(predict.fit.gini.test != test$good_bad)

#print
print(misClasificError.deviance.train)
print(misClasificError.deviance.test) #<- better results
print(misClasificError.gini.train)
print(misClasificError.gini.test)

#Assignment 2.3
```

```r
trainScore=rep(0,19)
testScore=rep(0,19)
for(i in 2:19) {
  prunedTree=prune.tree(fit.deviance,best=i)
  pred=predict(prunedTree, newdata=valid, type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}
plot(2:19, trainScore[2:19], type="b", col="red", ylim=c(0,600), ylab = "Deviance",
     xlab = "Tree length")
axis(side=1, at=c(2:19))
legend(2, 95, legend=c("Train", "Valid"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
points(2:19, testScore[2:19], type="b", col="blue")

best.n = match(c(min(testScore[2:19])), testScore)
print(paste("Best length:", best.n))
#number of leaves = 4
#the tree length with the lowest deviance is n=4
#selecting the tree n = 4
selected.tree=prune.tree(fit.deviance,best=best.n)
pred.selected.tree=predict(selected.tree, test, type="class")
plot(selected.tree)
text(selected.tree, pretty=0)

misClasificError.selected.tree = mean(pred.selected.tree != test$good_bad)
print(misClasificError.selected.tree) # this is 0.256

#Assignment 2.4
data = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab\ 2/credits
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]
id3=setdiff(id1,id2)
test=data[id3,]

library(MASS)
library(e1071)
naive.bayes.model=naiveBayes(as.factor(good_bad)~., data=train)
naive.bayes.model
naive.bayes.predict.train = predict(naive.bayes.model, newdata=train[,1:19])
naive.bayes.predict.test = predict(naive.bayes.model, newdata=test[,1:19])
#conf matrix

table(factor(train$good_bad, labels=c("Actual Bad", "Actual Good")), factor(naive.bayes.predict.train
, labels=c("Bad", "Good")))
table(factor(test$good_bad, labels=c("Actual Bad", "Actual Good")), factor(naive.bayes.predict.test
, labels=c("Bad", "Good")) )
```

```r
misClasificError.naive.bayes.predict.train = mean(naive.bayes.predict.train != train$good_bad)
misClasificError.naive.bayes.predict.test = mean(naive.bayes.predict.test != test$good_bad)
print(misClasificError.naive.bayes.predict.train)
print(misClasificError.naive.bayes.predict.test)

#Assignment 2.5
#type = "raw" vs "class":
#https://stackoverflow.com/questions/23085096/type-parameter-of-the-predict-function
naive.bayes.predict = predict(naive.bayes.model, newdata=test[,1:19], type = "raw")
naive.bayes.predict
pred.selected.tree.test=predict(selected.tree, test, type="vector")
pred.selected.tree.test
pred.selected.tree.roc = ifelse(pred.selected.tree.test[,2] > 0.5,1,0)
pred.selected.tree.roc

pi = seq(0.05, 0.95, by=0.05)
library(pROC)
naive.bayes.roc.vector.tpr = 1:length(pi)
naive.bayes.roc.vector.fpr = 1:length(pi)

dec.tree.roc.vector.tpr = 1:length(pi)
dec.tree.roc.vector.fpr = 1:length(pi)

for(i in 1:length(pi)){
  naive.bayes.roc = ifelse(naive.bayes.predict[,2] > pi[i],1,0)
  dec.matrix.bayes = as.matrix(table(as.factor(naive.bayes.roc), test$good_bad))
  dec.matrix.bayes
  #dec.matrix.bayes[2,2] = tp# dec.matrix.bayes[2,1] = fn
  naive.bayes.roc.vector.tpr[i] = dec.matrix.bayes[2,2]/(dec.matrix.bayes[1,2]+dec.matrix.bayes[2,2])
  naive.bayes.roc.vector.fpr[i] = dec.matrix.bayes[2,1]/(dec.matrix.bayes[2,1]+dec.matrix.bayes[1,1])

  pred.selected.tree.roc = ifelse(pred.selected.tree.test[,2] > pi[i],1,0)

  #if we get a unique vector
  if(length(unique(pred.selected.tree.roc)) > 1){
    dec.matrix.tree = as.matrix(table(as.factor(pred.selected.tree.roc), test$good_bad))
    dec.tree.roc.vector.tpr[i] = dec.matrix.tree[2,2]/(dec.matrix.tree[1,2]+dec.matrix.tree[2,2])
    dec.tree.roc.vector.fpr[i] = dec.matrix.tree[2,1]/(dec.matrix.tree[2,1]+dec.matrix.tree[1,1])

  }
}
plot(c(1,naive.bayes.roc.vector.fpr,0), c(1,naive.bayes.roc.vector.tpr,0), xlim=c(0, 1), ylim=c(0, 1),
     , type="b", xlab = "FPR", ylab = "TPR" )
#excluding points that did not write
points(c(dec.tree.roc.vector.fpr[dec.tree.roc.vector.fpr<=1],0)
       ,c(dec.tree.roc.vector.tpr[dec.tree.roc.vector.tpr<=1],0), col = "red", type="b")
abline(a=0, b=1)
legend(0.6, 0.2, legend=c("DT", "NB"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
#naive.bayes.predict = ifelse(naive.bayes.predict > pi,1,0)

#Assignment 2.6
data = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab\ 2/credits
```

```
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]
id3=setdiff(id1,id2)
test=data[id3,]

library(MASS)
library(e1071)
#loss matrix seems not to do anything
naive.bayes.model=naiveBayes(as.factor(good_bad)~., data=train, params = list(loss= matrix(c(0,10,1,0),
ncol = 2)))
naive.bayes.model
naive.bayes.predict.train = predict(naive.bayes.model, newdata=train[,1:19], type = "raw" )
naive.bayes.predict.test = predict(naive.bayes.model, newdata=test[,1:19], type = "raw" )

naive.bayes.predict.train = ifelse(1*naive.bayes.predict.train[,1]>10*naive.bayes.predict.train[,2]
, "bad","good")
naive.bayes.predict.test = ifelse(1*naive.bayes.predict.test[,1]>10*naive.bayes.predict.test[,2]
, "bad","good")

#conf matrix

table(factor(train$good_bad, labels=c("Actual Bad", "Actual Good")), factor(naive.bayes.predict.train,
                                                                            labels=c("Bad", "Good")))
table(factor(test$good_bad, labels=c("Actual Bad", "Actual Good")), factor(naive.bayes.predict.test,
labels=c("Bad", "Good")) )

misClasificError.naive.bayes.predict.train = mean(naive.bayes.predict.train != train$good_bad)
misClasificError.naive.bayes.predict.test = mean(naive.bayes.predict.test != test$good_bad)
print(misClasificError.naive.bayes.predict.train)
print(misClasificError.naive.bayes.predict.test)
```

Assignment 4

```
data = read.csv(file="/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab\ 2/NIRSpectra.csv"
                ,sep=";", dec=",")
set.seed(12345)

data1 = data
#creating matrix
data1$Viscosity = c()
res=prcomp(data1)
lambda=res$sdev^2
#eigenvalues
lambda
#proportion of variation sprintf("%2.3f",lambda/sum(lambda)*100) screeplot(res)
sprintf("%2.3f",lambda/sum(lambda)*100)
#components 1 and 2 capture over 99% of the total variance
```

```r
screeplot(res)

plot(res$x[,1], res$x[,2], ylab = "PC2", xlab = "PC1")
#res$x[,1][res$x[,1]>1]

#assignment 4.2
U=res$rotation

plot(U[ , 1 ], main="Traceplot, PC1", ylim=range(-0.5,0.5))
plot(U[ , 2 ],main="Traceplot, PC2", ylim = range(-0.5,0.5))

#Assignment 4.3

#setting data
S<-U
set.seed(12345)
#mixing matrix
A <-matrix(c(0.291, 0.6557, -0.5439, 0.5572), 2, 2)

plot(1:length(S[,1]), S[,1],xlab = "PC1", ylab = "", main="Traceplot, PC1", ylim = range(-0.5,0.5))
#par(new = TRUE)
plot(1:length(S[,2]), S[,2], xlab = "PC2", ylab = "", main="Traceplot, PC2", ylim = range(-0.5,0.5))

par(mfcol = c(1, 2))
plot(1:length(X[,1]), X[,1],xlab = "W'1", ylab = "", main="Traceplot, W'", ylim = range(-0.5,0.5))
plot(1:length(X[,2]), X[,2], xlab = "W'2", ylab = "", main="Traceplot, W'", ylim = range(-0.5,0.5))
library(fastICA)
#using ICA to estimate signals
a <-fastICA(data1, 2) #ICA

W.prime = a$K%*%a$W
plot(1:length(W.prime[,1]), W.prime[,1],xlab = "W'1", ylab = "", main="Traceplot, W'1",
     ylim = range(-13,3))
plot(1:length(W.prime[,2]), W.prime[,2],xlab = "W'2", ylab = "", main="Traceplot, W'2",
     ylim = range(-13,3))

plot(a$S[,1], a$S[,2])
```