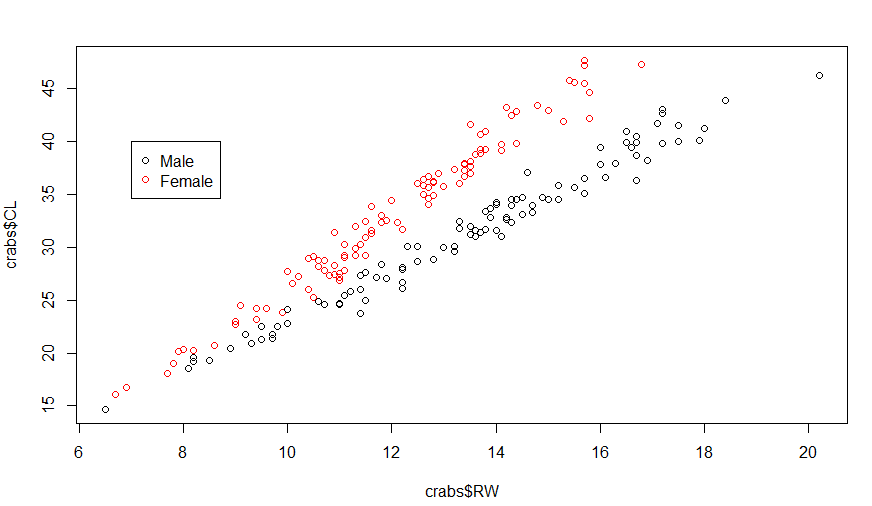
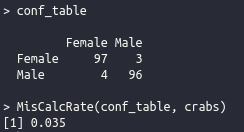
Assignment 1

1. **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:**

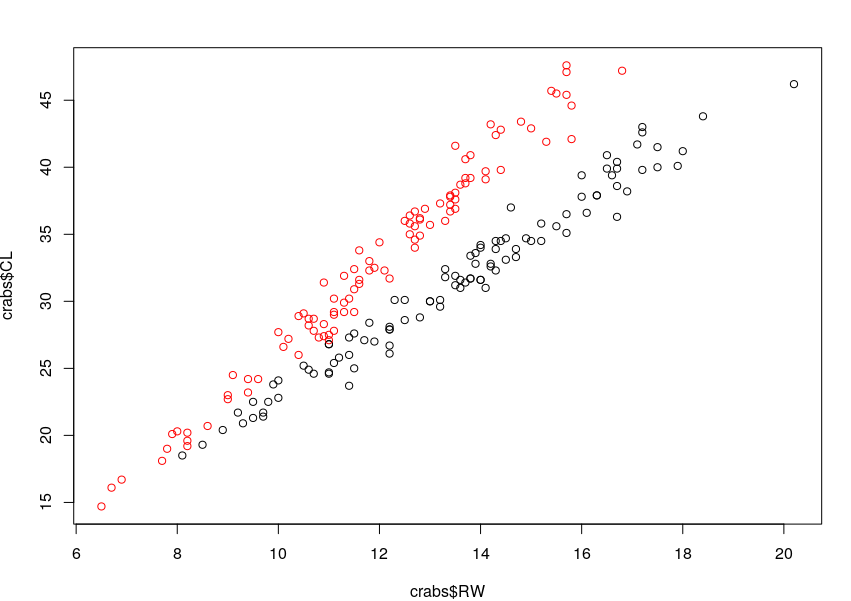
It is possible to make a LDA because there is a clear difference for the values between the different sexes of crabs.



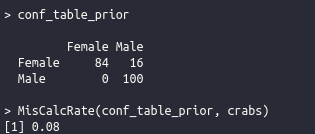
2. **Make LDA analysis with target Sex and features CL and RW and proportional prior by using lda() function in package MASS. Make a scatter plot of CL versus RW colored by the predicted Sex and compare it with the plot in step 1. Compute the misclassification error and comment on the quality of fit.**



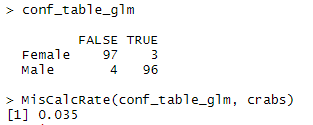
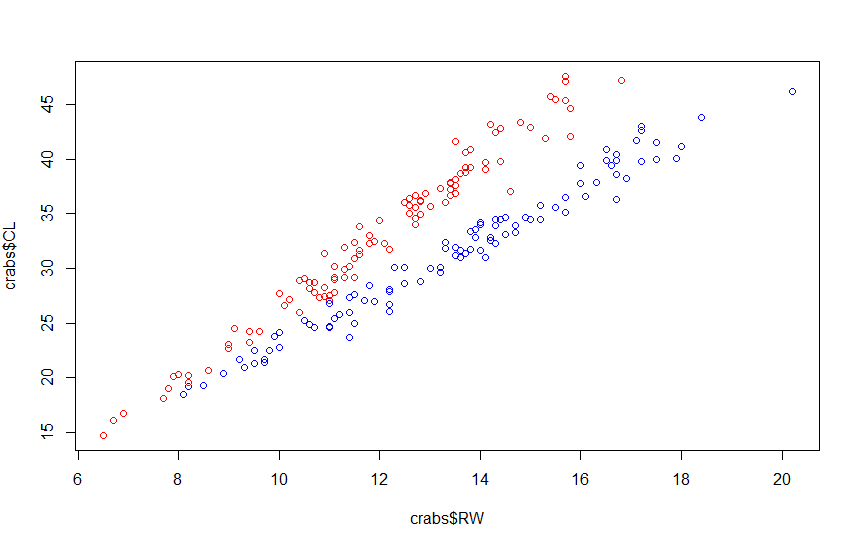
Below you can see the plotted predicted values and to the right you can see the confusion table and also the misclassification rate of 3.5%. This seems like a good value and validates the ability for a LDA to split the data through a linear boundary.



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

With a prior of p(Female)=0.1 and p(Male)=0.9 we get a lot more predicted Male’s in the outcome. This results in an increased misclassification rate to 8%. This is reasonable because setting a very high prior to the probability that a crab is a Male.

**4. Make a similar kind of classification by logistic regression (use function glm()), 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**

 Above we see the predicted values from the glm method, and to the right we see the confusion table where FALSE = Female and TRUE = Male. The misclassification rate is the same as in 2, 3.5%.



Above we see the classified data with the decision boundary of the equation:

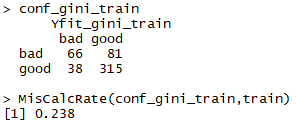
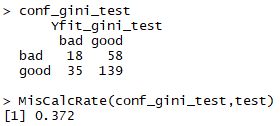
CL = -2,71\*RW – 2,96

Compared to the LDA results we see that it’s the exact same misclassification rate, but looking at the plots we see that there are differences in how the observations have been classified. The logistic regression made it easier for a boundary to be drawn by splitting the classificiation of observations more distinctly.

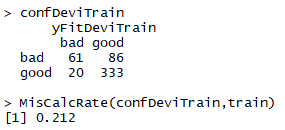
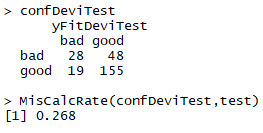
Assignment 2

1. **Fit a decision tree to the training data by using the following measures of impurity a. Deviance b. Gini index and report the misclassification rates for the training and test data. Choose the measure providing the better results for the following steps**

For the Gini index measurement the following Train and Test data confusion tables & misclass rates where obtained:

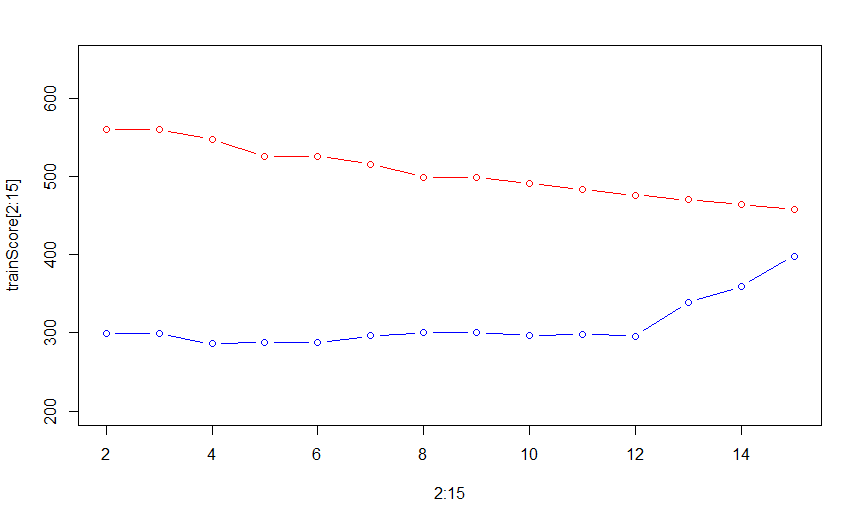
For the deviance measurement the following Train and Test data confusion tables & misclass rates where obtained:

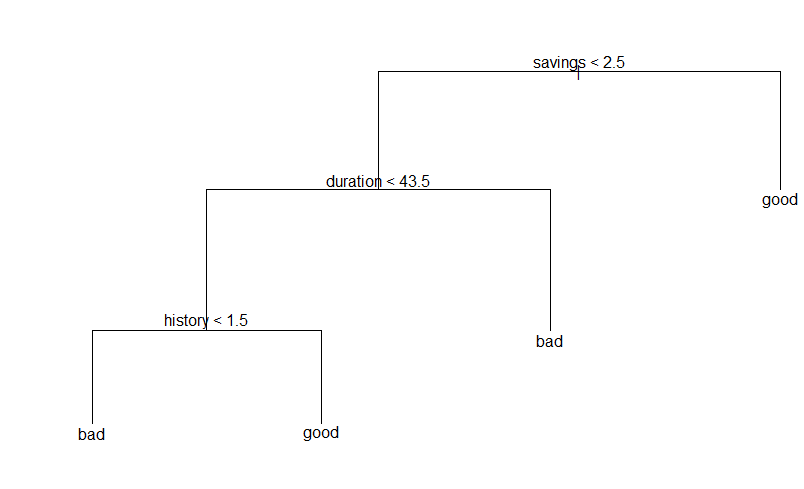
 

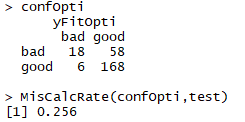
Since we got a lower miscalculation rate for the deviance measure, we will go with that one for the following steps.

1. **Use training and validation sets to choose the optimal tree depth. Present the graphs of the dependence of deviances for the training and the validation data on the number of leaves. Report the optimal tree, report it’s depth and the variables used by the tree. Interpret the information provided by the tree structure. Estimate the misclassification rate for the test data.**

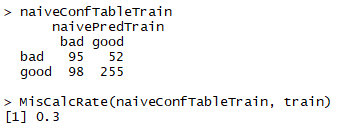
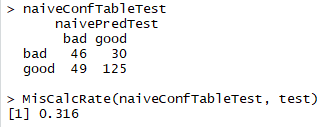
Below shows the graph how the deviance depends on the different number of leaves from 2..15. The optimal tree is the one with 4 leaves because it has the lowest deviance for the validation data.



The chosen optimal tree with four leaves is shown in graph below. The depth is equal to 3, as many as variables chosen, and the variables used are duration, history and savings.

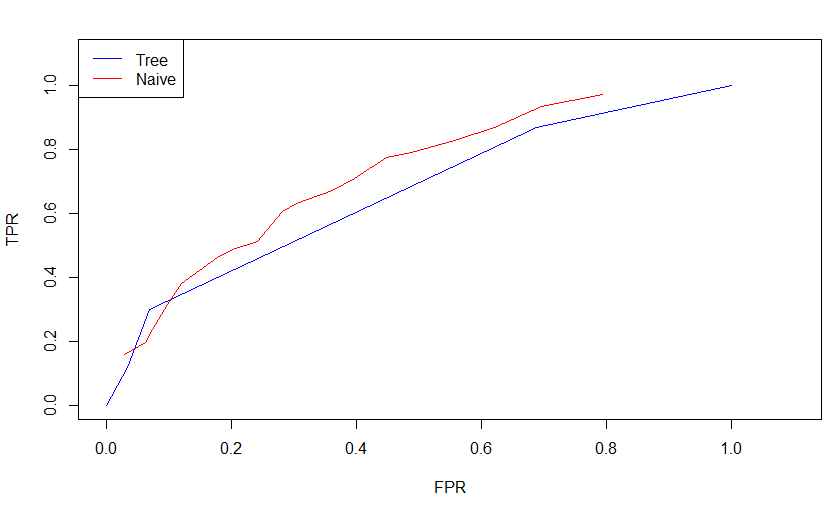


With this tree structure we get a misclassification rate of 25.6% for the test data. The tree structure tells us that a low savings rate is a negative sign causing us to analyze the duration variable, and if the duration is too high (<43.5) we classify as bad. Now if the duration is low we need to further investigate the history variable, with a low history value being bad and higher being able to classify as good.

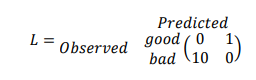
1. **Use training data to perform classification using Naïve Bayes and report the confusion matrices and misclassification rates for the training and for the test data. Compare the results with those from step 3.**

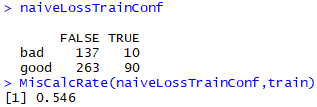
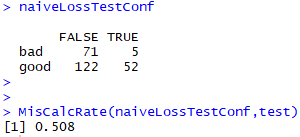
Above we see the confusion matrices and misclassification rates for training and test data with the naïve Bayes classification. Compared to the misclassification rate from step three this is higher and the tree structure seems like a better fit for this problem.

1. **Use the optimal tree and the Naïve Bayes model to classify the test data by using the following principle:**  **where** 𝜋 **= 0.05,0.1,0.15, … 0.9,0.95. Compute the TPR and FPR values for the two models and plot the corresponding ROC curves. Conclusion?**



Since there is more area under the curve (AUC) of the Naïve model on the ROC curve we can conclude that it is the best classifier according to ROC. Depending on what you prioritize different models can be preferred.

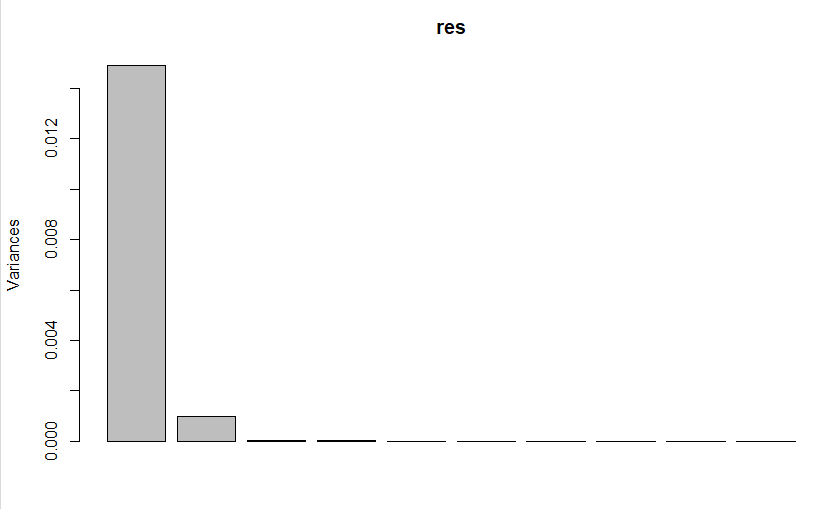
1. **Repeat Naïve Bayes classification as it was in step 4 but use the following loss matrix:**

**and report the confusion matrix for the training and test data. Compare the results with the results from step 4 and discuss how the rates has changed and why.**

We see a much higher misclassification rate than in step 4 because we punish a faulty prediction towards good 10 times harder, meaning that it is worse to give a loan to someone who should be classified as bad. This is why we classify many more good as bad but we have a much lower amount of good predictions that actually are bad. This is reasonable from the perspective of a bank, being more careful who you lend out money to.

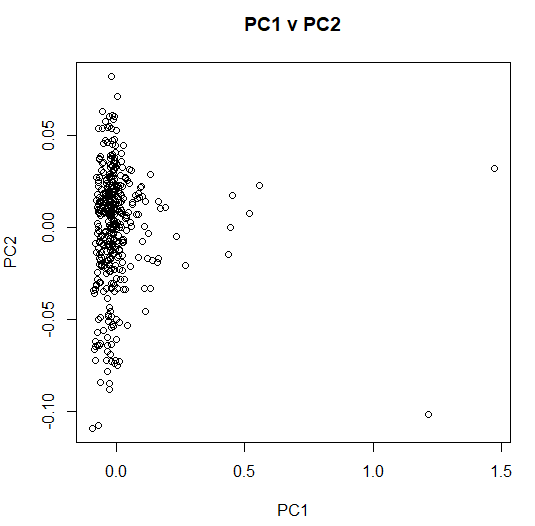
Assignment 4

1. **Conduct a standard PCA by using the feature space and provide a plot explaining how much variation is explained by each feature. Does the plot show how many PC should be extracted? Select the minimal number of components explaining at least 99% of the total variance. Provide also a plot of the scores in the coordinates (PC1, PC2). Are there unusual diesel fuels according to this plot?**

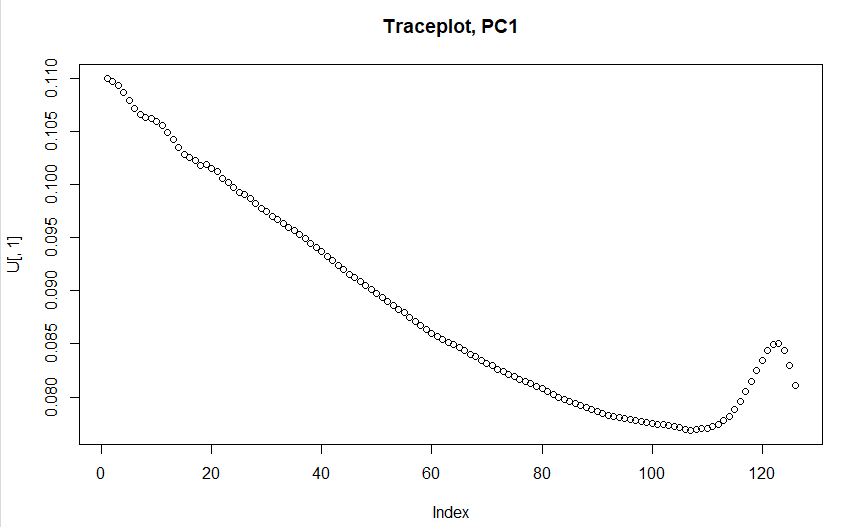
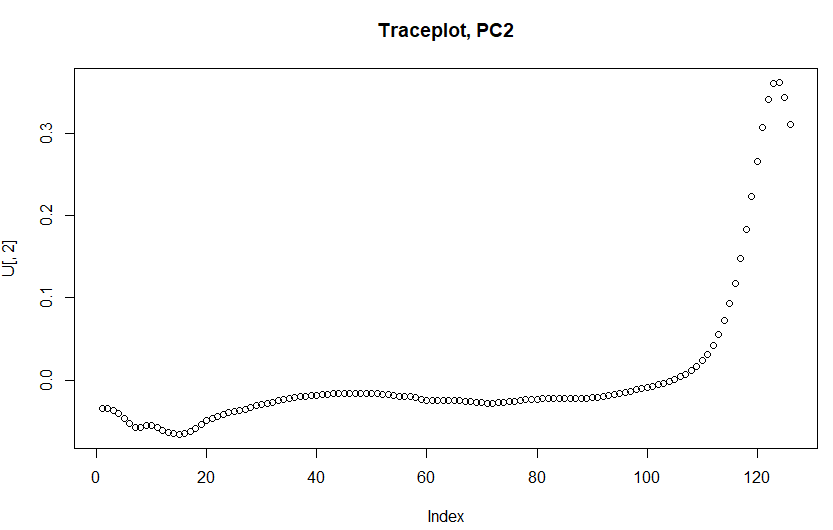


The histogram above shows that only PC1 and PC2 have a significant effect on the variance. This tells us that we need to extract these PCs.

PC1 captures 93.3% of the variance and PC2 captures 6.2%, giving us a total over 99% of the variance captured with these 2 PCs.



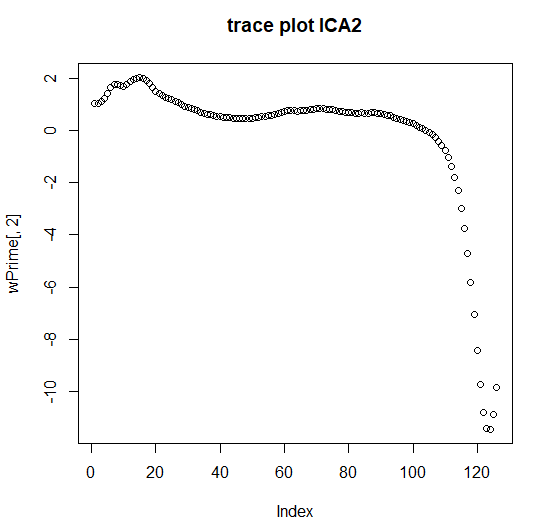
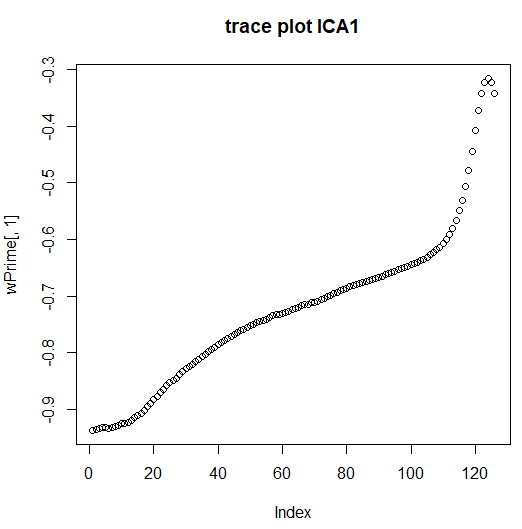
We have a couple of outliers with a big variance on PC1. A couple of points around PC1= 0.5 and 2 points above PC=1.

1. **Make trace plots of the loadings of the components selected in step 1. Is there any principle component that is explained by mainly a few original features?**

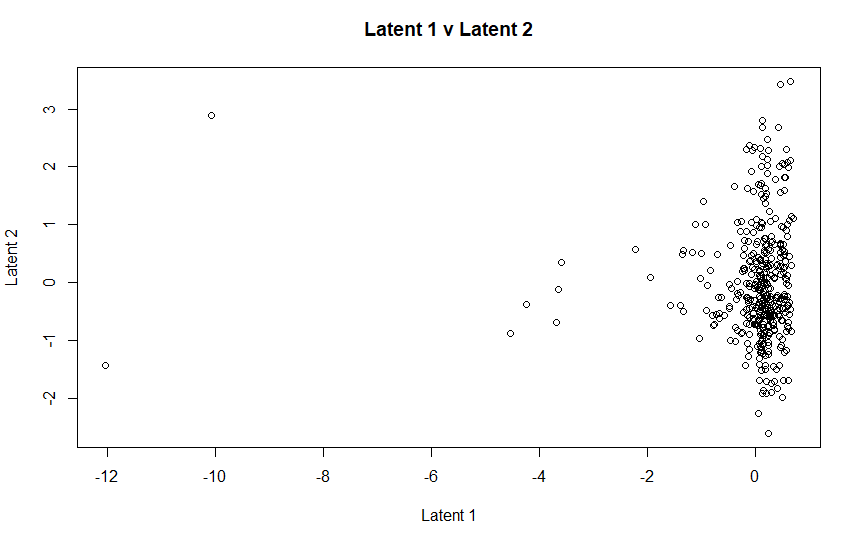
Most of the features have an impact on PC1 as we can see in the graph to the left. While most of the features do *not* have an impact on PC2 as most values for the different features are around 0. There are a couple of outliers that actually affect PC2 around feature number 115-126, as well as some in the beginning with values little below 0.

1. **Perform Independent Component Analysis with the number of components selected in step 1 (set seed 12345). Check the documentation for the fastICA method in R and do the following:**

**a. Compute 𝑊′ = 𝐾 ⋅ 𝑊 and present the columns of 𝑊′ in form of the trace plots. Compare with the trace plots in step 2 and make conclusions. What kind of measure is represented by the matrix 𝑊′ ?**

 Below we see the trace plots for the latent ICA components created from whitening the data by applying K (the pre-whitening matrix) onto W (our un-mixing matrix) to compute W’. This is done to “scale” the data. Compared with the trace plots in step 2 we see that the latent ICA component are mirrors of the PCA components. This is because PCA tries to find correlation between the features by maximizing the variance to allow reconstruction. ICA works maximizing independence through transforming the feature space into a new feature space where all features are mutually independent.

**b. Make a plot of the scores of the first two latent features and compare it with the score plot from step 1**



Just like in a. we see that also this is a mirror reflection of the scores from PCA but through performing ICA the features are scaled differently.

**Appendix**

**Assignment 1 CODE**

**crabs=read.csv("australian-crabs.csv")**

**n=dim(crabs)[1]**

**set.seed(12345)**

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

**plot(crabs$RW, crabs$CL, col=crabs$sex)**

**legend(7,40,unique(crabs$sex), col=1:length(crabs$sex),pch=1)**

**library(MASS)**

**lda\_sex<-lda(sex~CL+RW , data=crabs)**

**predict\_vals<-predict(lda\_sex , crabs)**

**predict\_vals**

**plot(crabs$RW, crabs$CL, col=predict\_vals$class)**

**MisCalcRate <- function(confusion\_table, inputdata){**

**n=length(inputdata[,1])**

**return(1-sum(diag(confusion\_table))/n)**

**}**

**conf\_table <- table(crabs$sex,predict\_vals$class)**

**conf\_table**

**MisCalcRate(conf\_table, crabs)**

**predict\_vals\_prior <- predict(lda\_sex, prior=c(0.1,0.9), crabs)**

**plot(crabs$RW, crabs$CL, col=predict\_vals\_prior$class)**

**conf\_table\_prior <- table(crabs$sex, predict\_vals\_prior$class)**

**conf\_table\_prior**

**MisCalcRate(conf\_table\_prior, crabs)**

**glm\_sex<-glm(sex~CL+RW, family=binomial,data=crabs)**

**predict\_glm <- predict(glm\_sex, crabs, type='response')**

**#male>0.5 = TRUE**

**plot(crabs$RW, crabs$CL, col=ifelse(predict\_glm>0.5,"red","blue"))**

**conf\_table\_glm <- table(crabs$sex,predict\_glm>0.5)**

**conf\_table\_glm**

**MisCalcRate(conf\_table\_glm, crabs) #0.035**

**install.packages("ggplot2")**

**library(ggplot2)**

**# func1 <-function(x) ((0.5-coef(glm\_sex[1])-x\*coef(glm\_sex[3]))/coef(glm\_sex[2]))**

**# plot(crabs$RW, crabs$CL, col=ifelse(predict\_glm>0.5,"red","blue"))**

**#coef(glm\_sex[1])+CL(y)\*coef(glm\_sex[2])+RW(x)\*coef(glm\_sex[3])=0.5**

**#ggplot**

**glmPlot <- ggplot(crabs, aes(x=RW,y=CL))**

**glmPlot <- glmPlot + geom\_point(aes(shape = factor(predict\_glm>0.5), colour=factor(predict\_glm>0.5)))**

**glmPlot <- glmPlot + geom\_abline(intercept = (-glm\_sex$coefficients[1]/glm\_sex$coefficients[2]), slope= (-glm\_sex$coefficients[3]/glm\_sex$coefficients[2]))**

**glmPlot**

**Assignment 2 CODE**

**RNGversion('3.5.1')**

**#url <- "https://cran.r-project.org/src/contrib/Archive/tree/tree\_1.0-39.tar.gz"**

**#pkgFile <- "tree\_1.0-39.tar.gz"**

**#download.file(url = url, destfile =pkgFile)**

**# #install.packages(pkgs=pkgFile, type="source", repos=NULL)**

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

**# library(tree)**

**# #install.packages("e1071")**

**# library(e1071)**

**# library(MASS)**

**creditscore=read.csv2("creditscoring.csv")**

**n=dim(creditscore)[1]**

**set.seed(12345)**

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

**train=creditscore[id,]**

**id1=setdiff(1:n, id)**

**set.seed(12345)**

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

**valid=creditscore[id2,]**

**id3=setdiff(id1,id2)**

**test=creditscore[id3,]**

**MisCalcRate <- function(confusion\_table, inputdata){**

**n=length(inputdata[,1])**

**return(1-sum(diag(confusion\_table))/n)**

**}**

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

**plot(fit\_tree\_gini)**

**text(fit\_tree\_gini,pretty=0)**

**#fit\_tree\_gini**

**summary(fit\_tree\_gini)**

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

**plot(fit\_tree\_devi)**

**text(fit\_tree\_devi,pretty=0)**

**#fit\_tree\_devi**

**summary(fit\_tree\_devi)**

**Yfit\_gini\_train <-predict(fit\_tree\_gini, train,type="class")**

**conf\_gini\_train <- table(train$good\_bad,Yfit\_gini\_train)**

**conf\_gini\_train**

**MisCalcRate(conf\_gini\_train,train)**

**Yfit\_gini\_test <-predict(fit\_tree\_gini, test,type="class")**

**conf\_gini\_test <- table(test$good\_bad,Yfit\_gini\_test)**

**conf\_gini\_test**

**MisCalcRate(conf\_gini\_test,test)**

**yFitDeviTrain <-predict(fit\_tree\_devi, train,type="class")**

**confDeviTrain <- table(train$good\_bad,yFitDeviTrain)**

**confDeviTrain**

**MisCalcRate(confDeviTrain,train)**

**yFitDeviTest <-predict(fit\_tree\_devi, test,type="class")**

**confDeviTest <- table(test$good\_bad,yFitDeviTest)**

**confDeviTest**

**MisCalcRate(confDeviTest,test)**

**trainScore= rep(0,15)**

**testScore= rep(0,15)**

**set.seed(12345)**

**for(i in 2:15) {**

**prunedTree=prune.tree(fit\_tree\_devi, best=i)**

**pred=predict(prunedTree, newdata=valid, type="tree")**

**trainScore[i]=deviance(prunedTree)**

**testScore[i]=deviance(pred)**

**}**

**plot(2:15, trainScore[2:15], type="b", col="red", ylim=c(200,650))**

**points(2:15, testScore[2:15], type="b", col="blue")**

**#optimal tree 4 leaves for test data**

**optimalTree <- prune.tree(fit\_tree\_devi, best=4)**

**yFitOpti <- predict(optimalTree, newdata=test, type="class")**

**confOpti <- table(test$good\_bad, yFitOpti)**

**confOpti**

**MisCalcRate(confOpti,test)**

**plot(optimalTree)**

**text(optimalTree, pretty=0)**

**yFitOptiP <- predict(optimalTree, newdata=test, type="vector")**

**naiveFit <- naiveBayes(good\_bad~. , data=train)**

**#print(naiveFit)**

**naivePredTrain <- predict(naiveFit, newdata=train)**

**naivePredTest <- predict(naiveFit, newdata=test)**

**naiveConfTableTrain <- table(train$good\_bad, naivePredTrain)**

**naiveConfTableTest <- table(test$good\_bad, naivePredTest)**

**naiveConfTableTrain**

**MisCalcRate(naiveConfTableTrain, train)**

**naiveConfTableTest**

**MisCalcRate(naiveConfTableTest, test)**

**set.seed(12345)**

**yFitOptiP <- predict(optimalTree, newdata=test, type="vector")**

**naiveFitRaw <- predict(naiveFit, newdata = test, type="raw")**

**fprTree = c()**

**tprTree = c()**

**fprNaive = c()**

**tprNaive = c()**

**#compare prediction in optimal tree to seq and then compare to realvalues for FPS and TPR**

**for(i in seq(0.05,0.95,0.05)){**

**curVal = c()**

**for(p in 1:250){**

**if(yFitOptiP[p,2]>i){ #ska vÃ¤lja good och ska vÃ¤lja 1-250 vÃ¤rden**

**curVal =c(curVal, 1)**

**}else {**

**curVal = c(curVal, 0)**

**}**

**}**

**tempTable <- table(valid$good\_bad, curVal)# spara värde och gör funktioner för FPR och TPR spara i numerisk vektor 1-20**

**if(dim(tempTable)[2] == 1){**

**if(colnames(tempTable) == "0"){**

**tempTable <- cbind(tempTable, c(0,0))**

**} else {**

**tempTable <- cbind(c(0,0), tempTable)**

**}**

**}**

**if(dim(tempTable)[2] == 2){**

**print(tempTable)**

**tprTree= c(tprTree, tempTable[2,2]/(tempTable[2,2]+tempTable[2,1]))**

**fprTree= c(fprTree, tempTable[1,2]/(tempTable[1,2]+tempTable[1,1]))**

**}**

**}**

**#compare prediction in naive to seq and then compare to realvalues for FPS and TPR**

**for(i in seq(0.05,0.95,0.05)){**

**curVal = c()**

**for(p in 1:250){**

**if(naiveFitRaw[p,2]>i){ #ska vÃ¤lja good och ska vÃ¤lja 1-250 vÃ¤rden**

**curVal =c(curVal, 1)**

**}else {**

**curVal = c(curVal, 0)**

**}**

**}**

**tempTable <- table(test$good\_bad, curVal) # spara värde och gör funktioner för FPR och TPR spara i numerisk vektor 1-20**

**if(dim(tempTable)[2] == 2){**

**tprNaive = c(tprNaive, tempTable[2,2]/rowSums(tempTable)[2])**

**fprNaive = c(fprNaive, tempTable[1,2]/rowSums(tempTable)[1])**

**}**

**}**

**fprTree<-sort(fprTree)**

**tprTree<-sort(tprTree)**

**# fprNaive<-sort(fprNaive)**

**# tprNaive<-sort(tprNaive)**

**plot(fprTree,tprTree, xlab="FPR", ylab="TPR", col="blue", xlim=c(0,1.1), ylim=c(0,1.1), type="l")**

**points(fprNaive,tprNaive,col="red",type="l")**

**legend("topleft",legend=c("Tree", "Naive"), col=c("blue", "red"),lty=c(1,1))**

**lossMatrix <- matrix(c(0, 1,10, 0), 2, 2)**

**naiveFit <- naiveBayes(good\_bad~. , data = train)**

**naiveLossTrain <- predict(naiveFit, newdata = train,type="raw")**

**naiveLossTest <- predict(naiveFit, newdata = test,type="raw")**

**naiveLossTrainConf <- table(train$good\_bad, naiveLossTrain[,2]/naiveLossTrain[,1] > 10)**

**naiveLossTestConf <- table(test$good\_bad, naiveLossTest[,2]/naiveLossTest[,1] > 10)**

**naiveLossTrainConf**

**MisCalcRate(naiveLossTrainConf,train) #0.546**

**naiveLossTestConf**

**MisCalcRate(naiveLossTestConf,test) #0.508**

**Assignment 4 Code**

**# #install.packages("fastICA")**

**# library(fastICA)**

**# #install.packages("pls")**

**# library(pls)**

**RNGversion('3.5.1')**

**NIR <- read.csv2("NIRspectra.csv")**

**NIR$Viscosity <- c()**

**res=prcomp(NIR)**

**lambda <- res$sdev^2**

**#eigenvalues**

**lambda**

**#displays variation captured of each feature**

**sprintf("%2.3f",lambda/sum(lambda)\*100)**

**#plot histograms of variance**

**screeplot(res)**

**#scores of PC1 & PC2 coordinates**

**plot(res$x[,1], res$x[,2], main = "PC1 v PC2", xlab = "PC1", ylab = "PC2")**

**U <- res$rotation**

**plot(U[,1], main="Traceplot, PC1")**

**plot(U[,2],main="Traceplot, PC2")**

**#ICA**

**set.seed(12345)**

**ICA <- fastICA(NIR, 2)**

**wPrime <- ICA$K %\*% ICA$W**

**plot(wPrime[,1], main="trace plot ICA1")**

**plot(wPrime[,2], main="trace plot ICA2")**

**plot(ICA$S[,1], ICA$S[,2], main = "Latent 1 v Latent 2", xlab ="Latent 1", ylab="Latent 2")**