```
#### Exam 2 - Assignment 1
### Libraries
library(e1071)
library(pls)

### Setup
data.crabs = read.csv("australian-crabs.csv")
data.blue = data.crabs[data.crabs$species=="Blue",]
data.oran = data.crabs[data.crabs$species=="Orange",]


### Functions

## Returns the confusion matrix
confusion_matrix = function(y, yfit) {
  return(table(y, yfit, dnn=c("TRUE", "PRED")))
}

## Returns the missclassifcationrate
mcr = function(cm){
  return(1-sum(diag(cm))/sum(cm))
}


### Implementation
## Task 1
plot(data.blue$BD, data.blue$CW, col="blue")
points(data.oran$BD, data.oran$CW, col="orange")
# Comment:
# Yes, BD and CW are reasonable predictors of the specie
# Since, the data points are separated on the specie,
# However, the classification line will have a harder time to
# predict correctly among the middle portion of BD and CW since
# those datapoints are overlapping some

## Task 2
naive.model = naiveBayes(species~BD+CW, data=data.crabs)
yfit.naive = predict(naive.model, data.crabs)
cm.naive = confusion_matrix(data.crabs$species, yfit.naive)
mcr.naive = mcr(cm.naive) # 0.395
# Comment:
# The quality is not that high with a missclassifcation rate ~ 40%
# Naive bayes is implemented under the assumption that there is a
strong
# (naive) independence between the features, which we've seen in the
graph
# From Task 1 that BD and CW are not.
# A linear classifier would be more appropriate to classify the data.

## Task 3
logistic.model = glm(species~BD+CW, data=data.crabs,
family="binomial")
yfit.logistic= ifelse(predict(logistic.model, data.crabs,
type="response")>0.5, 1, 0)
```

```
cm.logistic = confusion_matrix(data.crabs$species, yfit.logistic)
mcr.logistic = mcr(cm.logistic) # 0.0.02
coef = logistic.model$coefficients
boundry.logistic = -(coef[1]+coef[2]*data.crabs$BD)/coef[3]
plot(data.crabs$BD, data.crabs$CW, col=yfit.logistic+1,
     main="CW vs. BD", xlab="BD", ylab="CW")
lines(data.crabs$BD, boundry.logistic)

## Task 4
scaled.CW = scale(data.crabs$CW)
scaled.BD = scale(data.crabs$BD)
data.scaled = data.frame(CW=scaled.CW, BD=scaled.BD)
PCA = prcomp(data.scaled)
λ = PCA$sdev^2
variation.proportion = λ/sum(λ)*100
barplot(variation.proportion[1:2], ylim=c(0,100), col="forestgreen",
        main="Variation explanaition proportions for different
eigenvalues",
        xlab="PCi", ylab="Variation proportion")
# Comment:
# The reson that PC1 contains to much variation is because of the
shape of the original data
# With PCA being based upon maximizing the variance in the
projections:
# The shape is very long but not so thick, with PC1 explaining
variation in the longness of the shape
# whereas the variation in the thickness is explained by PC2.
## The equations expressing principal component coordinates through
the original coordinates:
#Rotation:
#         PC1        PC2
#CW 0.7071068 -0.7071068
#BD 0.7071068  0.7071068
# Rotaion = U
# [scaled.CW, scaled.BD] = X
# -> PC = X %*% U
X = as.matrix(data.scaled)
plot(X[,1], X[,2])
U = PCA$rotation
PC = X %*% U
plot(PC[,1], PC[,2])
plot(PCA$x[,1], PCA$x[,2])
# Both are the same
X = PC %*% solve(U)
plot(X[,1], X[,2])

## Task 5 -
data.crabs.PCA = data.frame(PCA$x, species=data.crabs$species)
naive.model.PCA = naiveBayes(species~PC1+PC2, data=data.crabs.PCA)
yfit.naive.PCA = predict(naive.model.PCA, data.crabs.PCA)
cm.naive.PCA = confusion_matrix(data.crabs.scaled$species,
yfit.naive.PCA)
mcr.naive.PCA = mcr(cm.naive.PCA) # 0.05
# Comment:
```

```
# The projections of the PCs are orthogonal and thus the independence
assumption is
# better "lived up to" and thus the naive bayesian classification
performs a lot better 0.05<<0.4
```