Lab 2 - Anton Gefvert

Anton Gefvert

12/3/2018

Assignment 1

1.1

Looking at Figure 1, we see that drawing a line to discriminate between male and female crabs on these data would be very feasable, thus classifying by linear discriminant analysis would be easy in this case.

1.2

If we compare Figure 2 to Figure 1, we see that they are very similar. There are some differences when carapace length is small, e.g the bottom left point is actually female. but is classified as male with the lda. There are also some differences when you look at the points which are kind of overlapping each other (some males classfied as females and vice versa). We notice though, that the bigger the carapace length is, the more accurate the lda model is!

When using this lda model we get a missclassification rate of 4.5%, this indicates that the model is very well fitted to this problem.

1.3

As seen in Figure 3,when using $p(Male) = 0.9, p(Female) = 0.1$ as a prior, we get (compared to Figure 3), as expected when weighting the male sex higher, a more male dominated graph. This can especially be seen in the lower bounds of carapace length and the values that are pretty close in between the two separation (e.g. the value around $CL = 38$ and $RW = 15$ is female in Figure 2 and male in Figure 3).

If we look at the missclassification rate when using the prior we get a missclassification rate of 8%, this is still very good, but almost twice as much when not using prior (or rather using a prior $p(Male) = p(Female) = 0.5$)

1.4

If we look at Figure 4 we see that it looks very similar to all the other figures. It looks very much like Figure 2, and is more keen to classify uncertain values as female than when using lda with prior.

Missclassification rate for this method is 3.5%, so slightly better than the other methods.

The equation for the decision boundary (green line in Figure 4) is $RW = 0.369CL + 1.08$

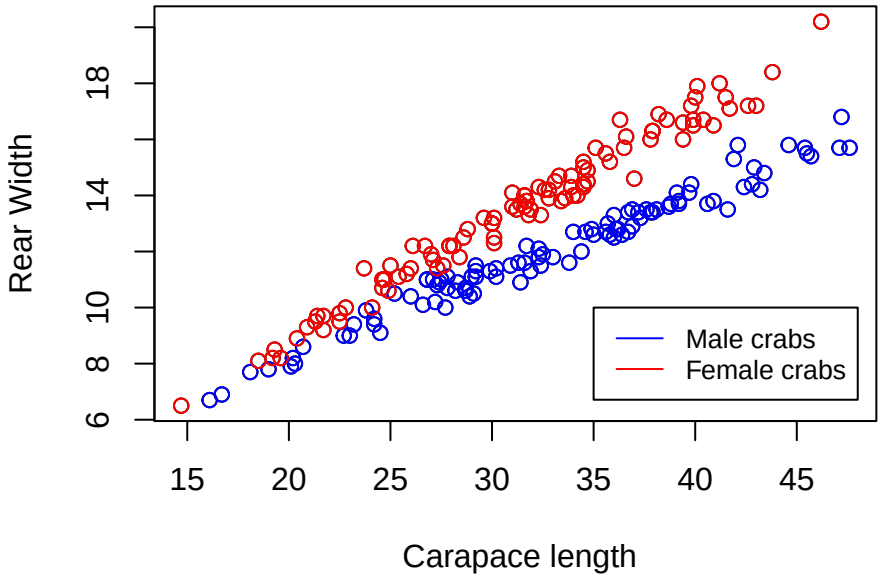


Figure 1: Carapace Length vs Rear Width classified by sex

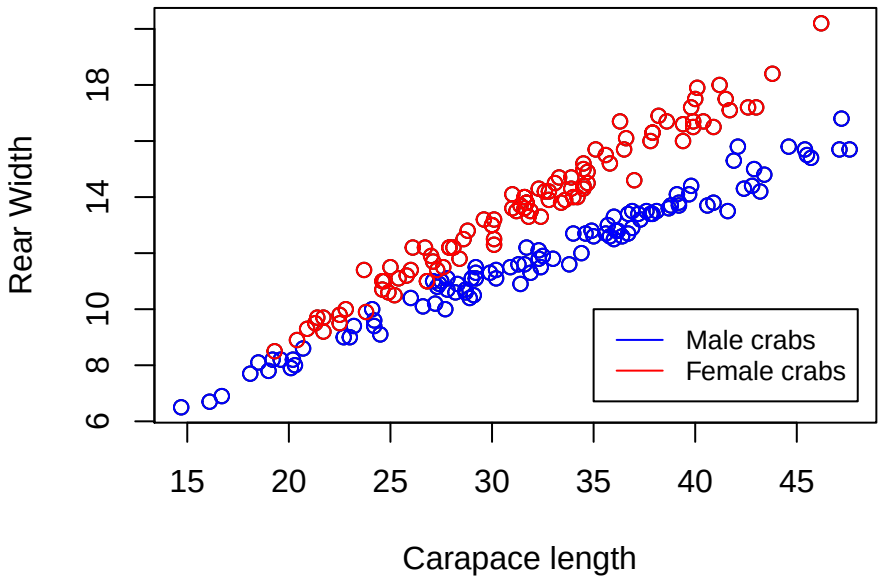


Figure 2: Carapace Length vs Rear Width classified by sex using lda

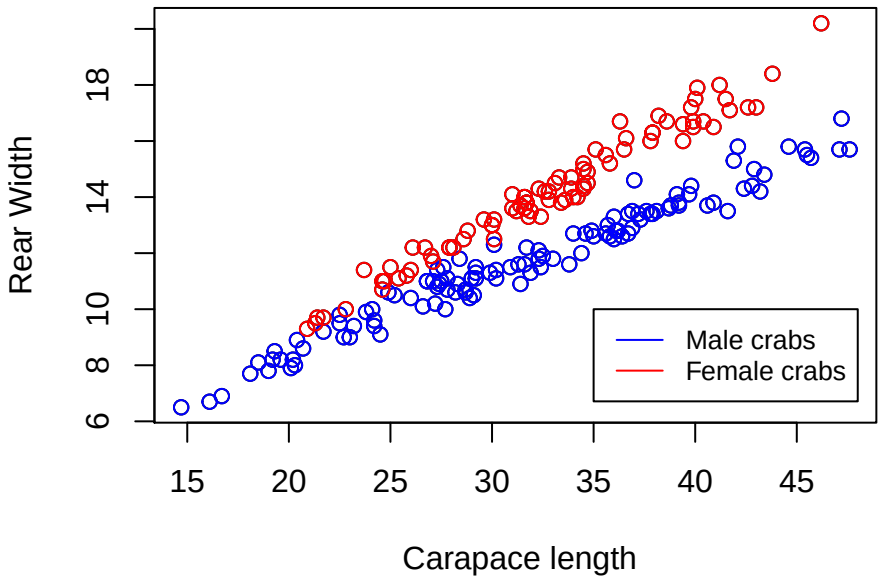


Figure 3: Carapace Length vs Rear Width classified by sex using lda (with prior)

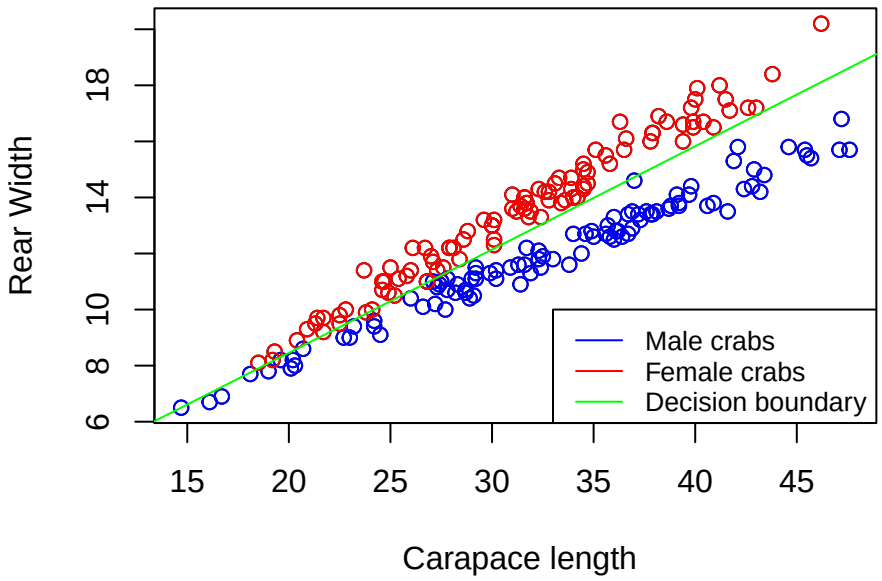


Figure 4: Carapace Length vs Rear Width classified by sex using glm

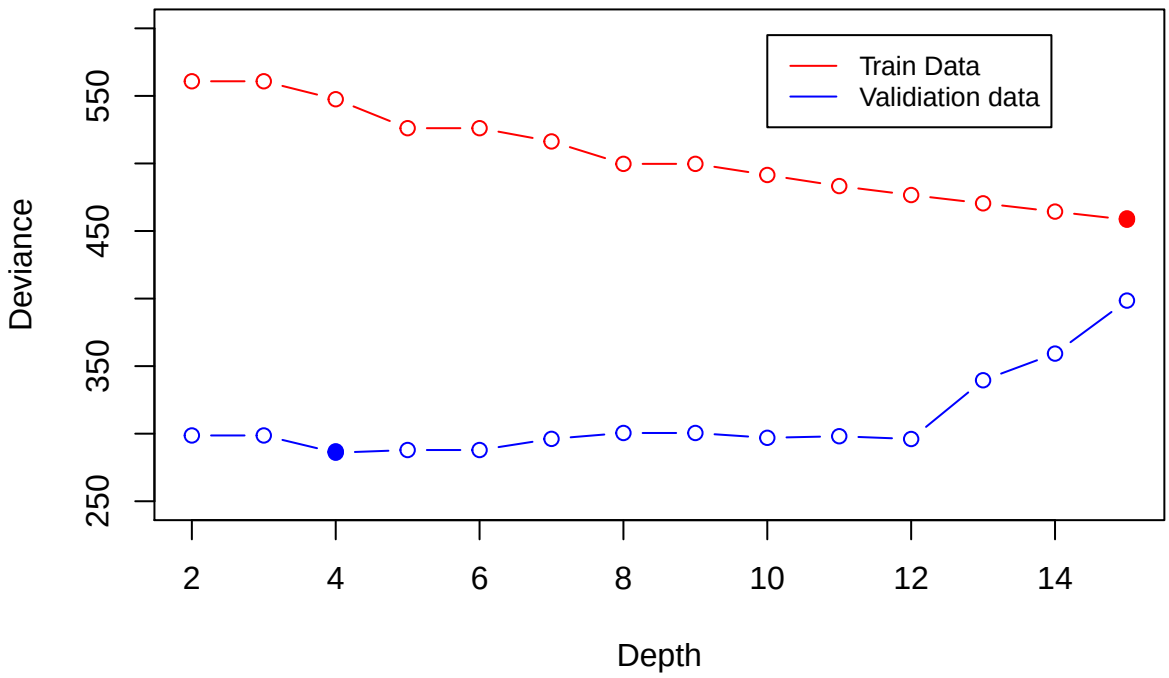


Figure 5: Graph of deviance over depth in decision tree

Assignment 2

2.2

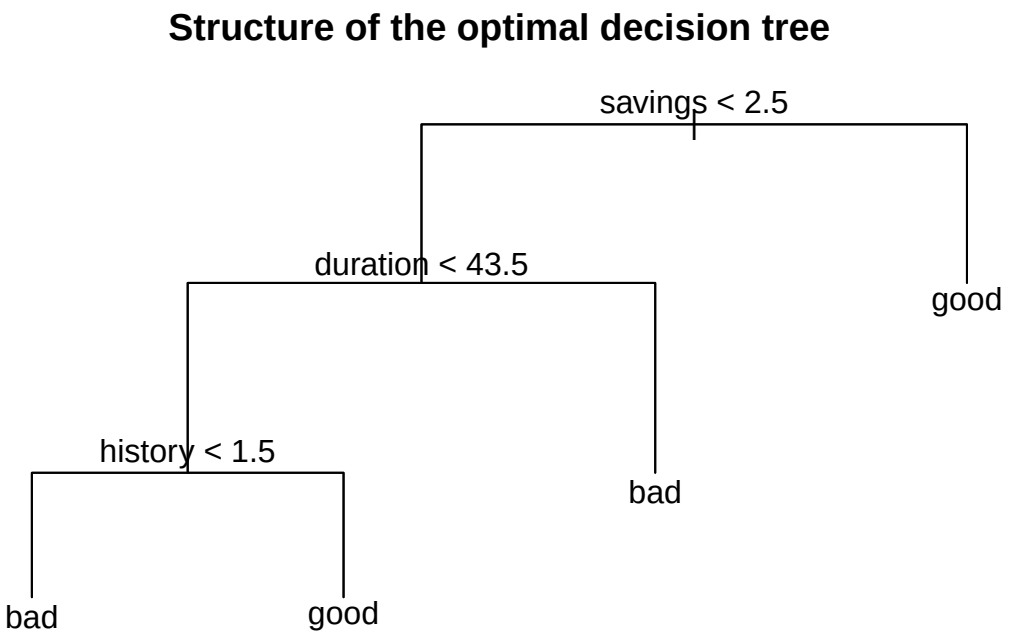
The missclassification rates when using decision trees with different impurity measures are as following:

- Deviance
  - Train: 21.2%
  - Test: 26.8%
- Gini index
  - Train: 24.2%
  - Test: 37.2%

As we can see, the Deviance is better in both train and test, thus we select this as the measure for following tasks!

2.3

If we look at Figure 5, we see that training data gets better and better the higher the depth, with the best node at $depth = 15$(maximum depth, denoted by filled red point). This is definetly overfitting if we compare it to the validiation data, which gets worse with higher depth. As we can see in the graph the optimal depth for validation data is $depth = 4$ (denoted by filled blue point)



As seen in figure above, describing the tree, the variables used are *savings*, *duration* and *history*. If we look at the graph, it looks like the chance of someone paying their loan back is high if they either have a lot of saving or if it's a short loan and they a have a good credit history.

Table 1: Confusion matrix for decision tree on test data

	bad	good
bad	18	58
good	6	168

In Table 1 we see the confusion matrix for the optimal decision tree on test data. This gives us a missclassification rate of 25.6%

2.4

Table 2: Confusion matrix for naive bayes on training data

	bad	good
bad	95	52
good	98	255

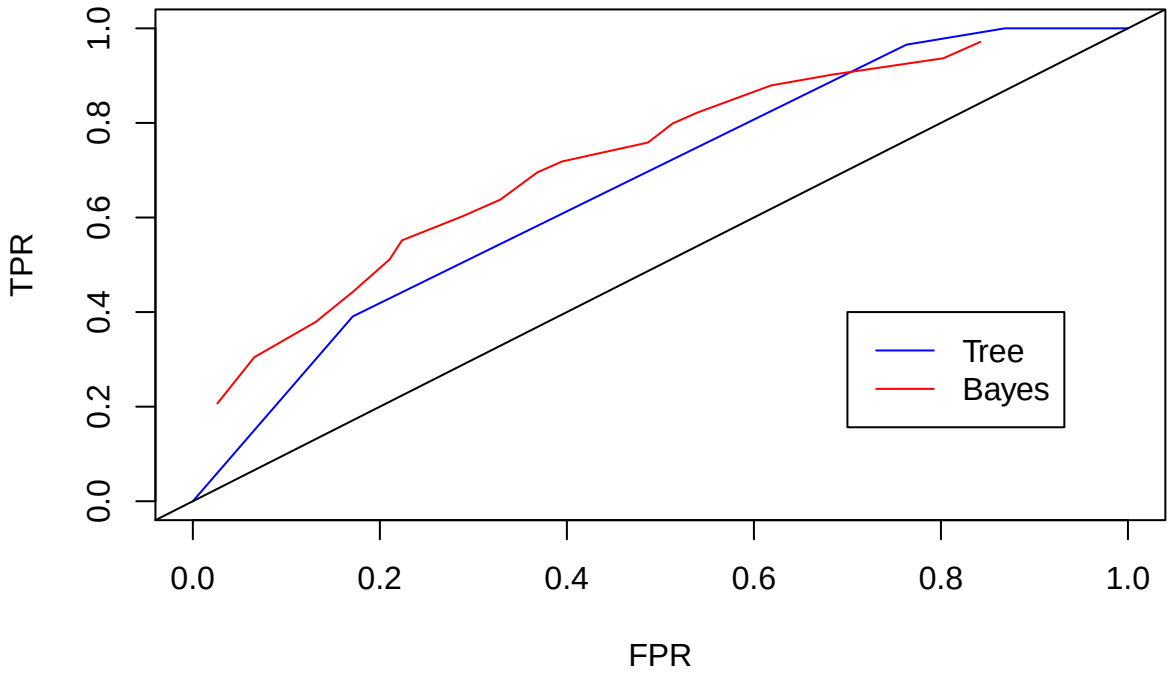


Figure 6: ROC curve for decision tree and naive bayes

Table 3: Confusion matrix for naive bayes on test data

	bad	good
bad	46	30
good	49	125

As we seen in Table 2 and Table 3, we have the confusion matrices for naive bayes. This gives us the classification rates:

- Train: 30%
- Test: 35.6%

If we compare this to the decision tree missclassification rate of 25.6% on test data, we see that the naive bayes performs worse in this case.

2.5

If we look at Figure 6, we see that the curve for naive bayes is above the curve for the decision tree, thus we can conclude that the naive bayes method is better in this case.

2.6

Table 4: Confusion matrix for naive bayes on train data using loss matrix

	bad	good
bad	137	10
good	263	90

Table 5: Confusion matrix for naive bayes on test data using loss matrix

	bad	good
bad	71	5
good	122	52

Table 4 and Table 5 shows the confusion matrix for training and test data respsectively, with naive bayes method when using the loss matrix $\begin{matrix} \\ bad \\ good \end{matrix}\begin{pmatrix} 0 & 10 \\ 1 & 0 \end{pmatrix}$. We can see that when using this method, we are more keen to predict bad, since we rather avoid giving someone a loan, than giving a loan to someone who will not pay back. This is because we weight the "predict=bad,actual=good" higher than "predict=good,actual=bad" using the loss matrix, meaning our model has to be very certain for it to predict "good".

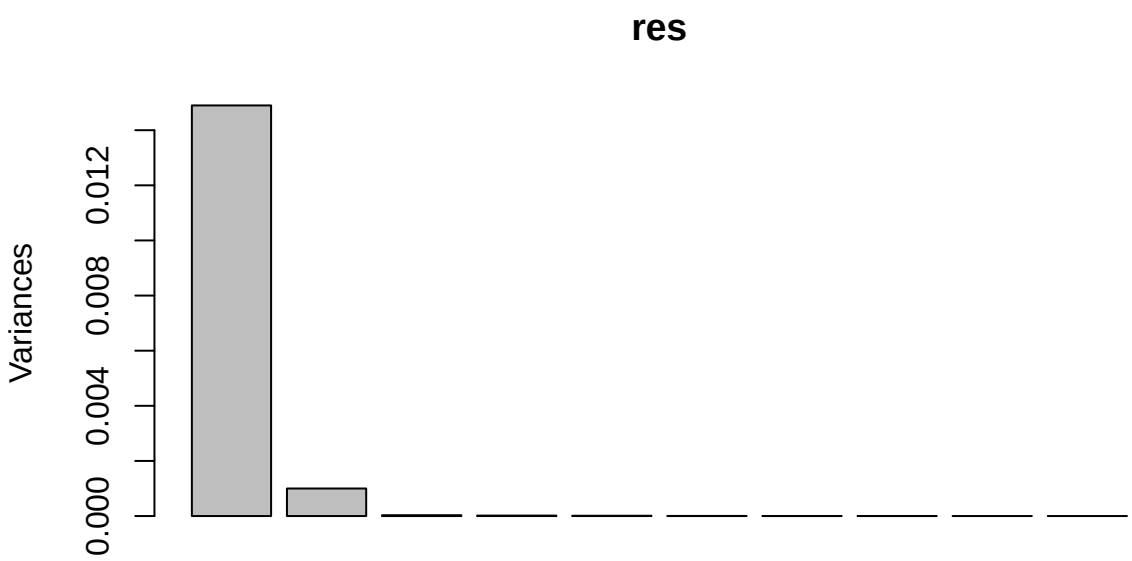


Figure 7: PCA plot of diesel data

Assignment 4

4.1

As seen in Figure 7, the minimum amount of components needed to get at least 99% of the total variance is 2.

As seen in Figure 8, there are a few unusual diesel fuels (indicated by red dots) if we compare these to the seemingly big cluster around 0.

4.2

If we look at Figure 10, we see that most of the values are hovering around 0, meaning they don't really matter for this component (PC2), there are also some values (at the end of the graph) with a high absolute values. This implies PC2 has only a few features describing it as a component. If we compare this to Figure 9, we see that this component (PC1) has values hovering around 0.09 for each feature, meaning they all matter to this component.

4.3

4.3a)

If we compare Figure 11 with Figure 9, we see that these are not very similar, but if we invert one of them, they look reasonably similar, but with a different scale. If we compare Figure 12 with Figure 10, we see that

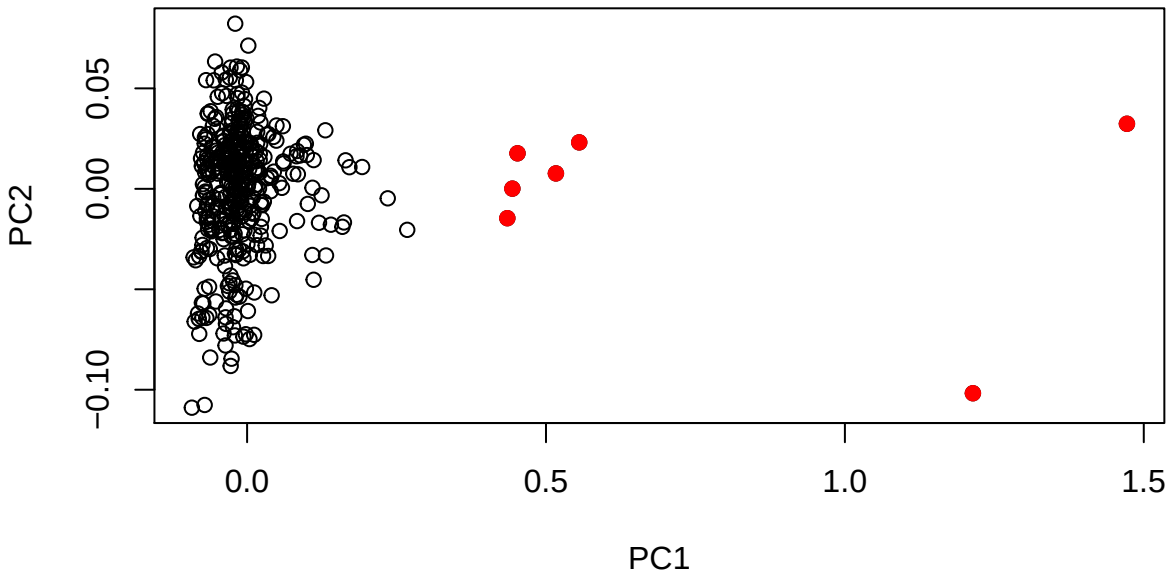


Figure 8: Graph of PC2 over PC1 on diesel data

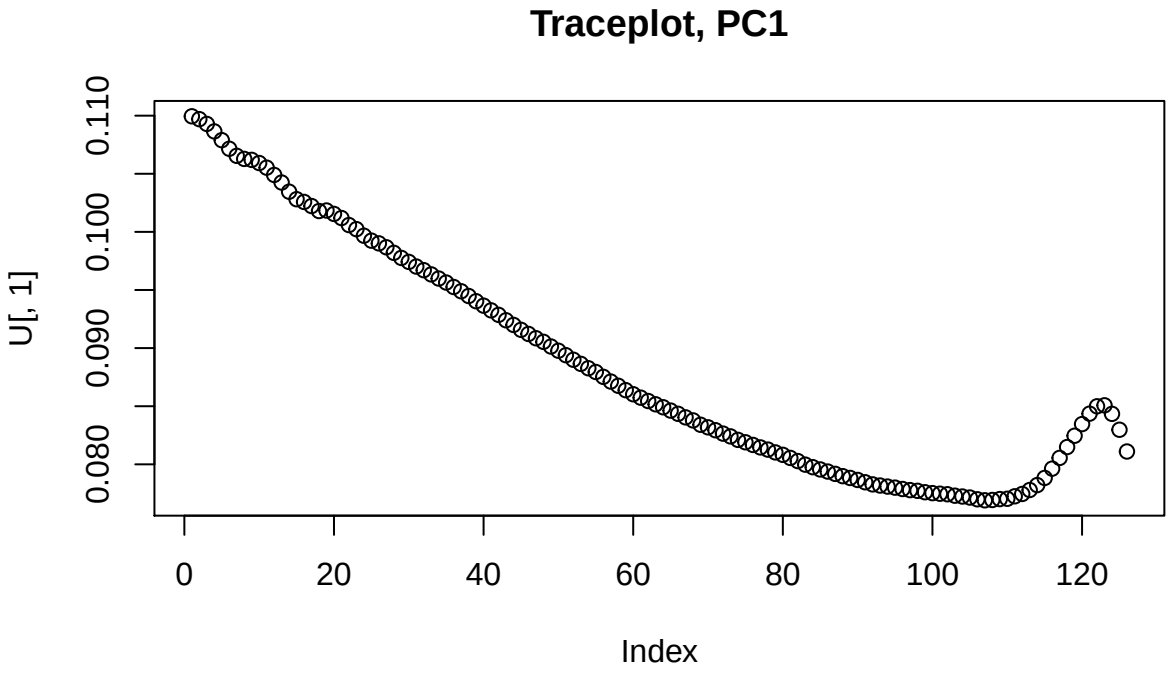


Figure 9: Traceplot for PC1

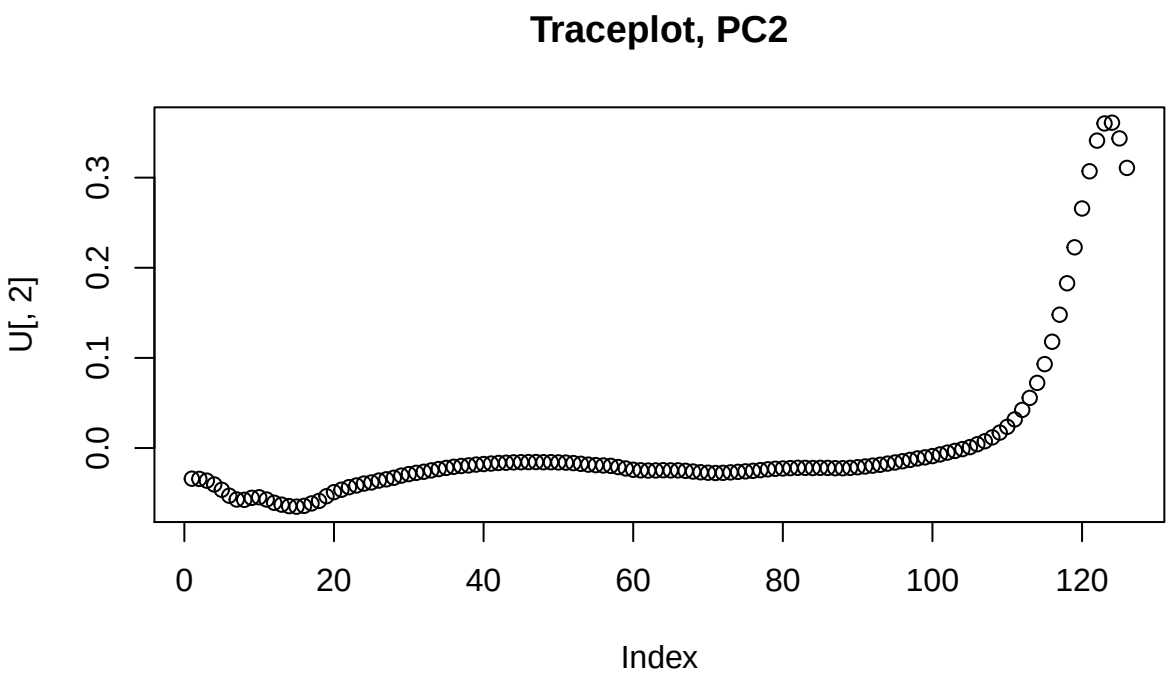


Figure 10: Traceplot for PC2

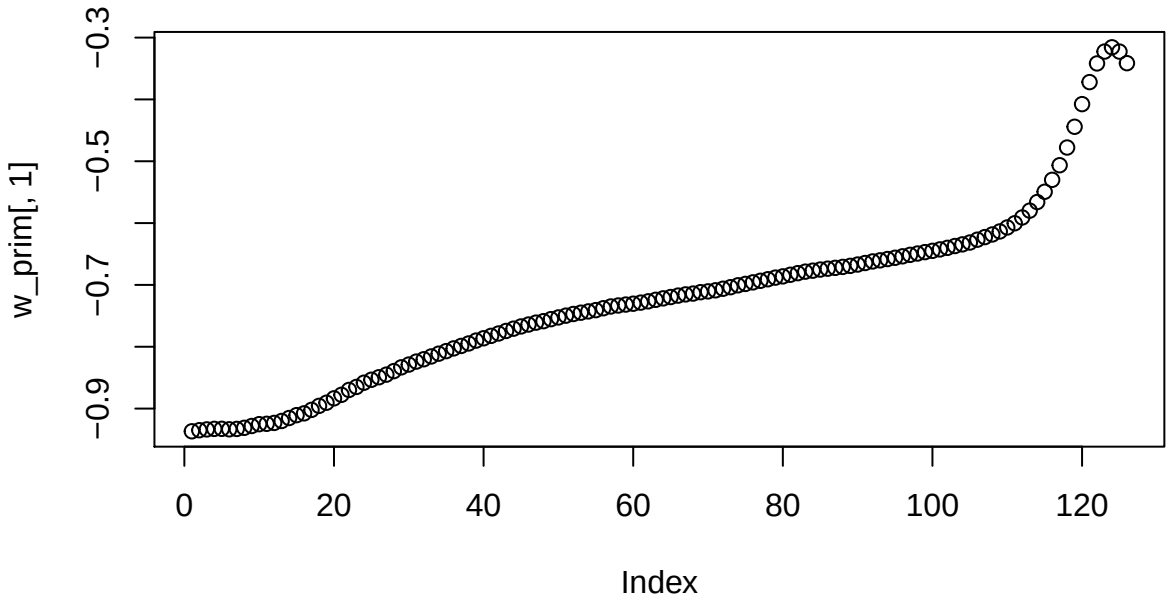


Figure 11: , Traceplot for PC1 using ICA

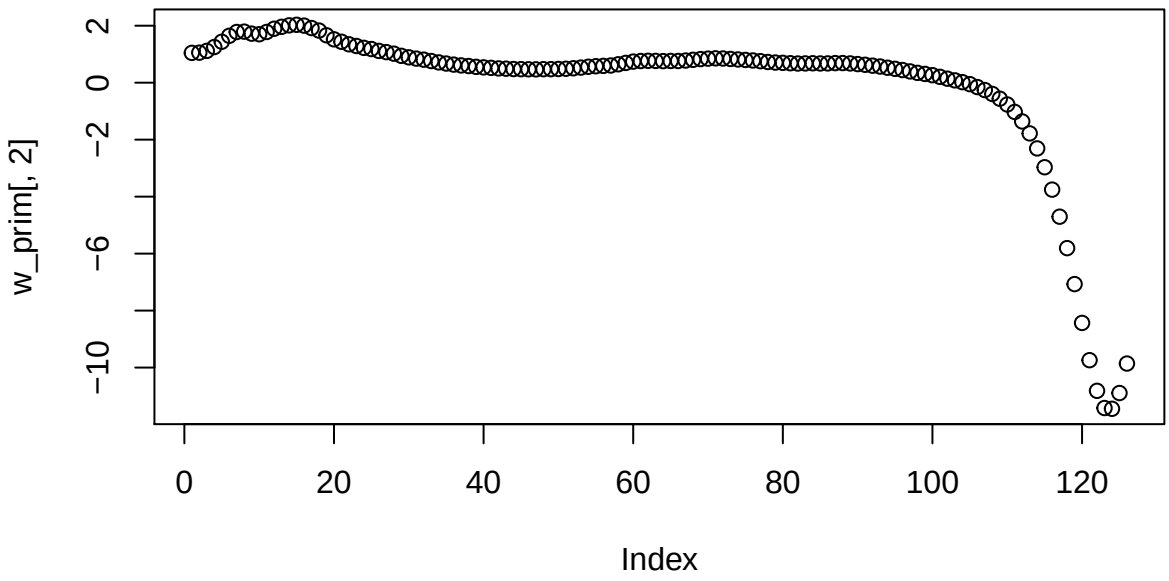


Figure 12: , Traceplot for PC2 using ICA

the shape of these trace plots are very similar but inverted, although the values on the y-axis are not the same, this could be due to some normalization or equivalent.

The measure represented by W' is the loadings for the ICA.

4.3b)

If we compare Figure 13 to Figure 8, we see that it looks like a rotated version of the PCA components, where the line that used to be the x-axis is now rotated 45 degrees.

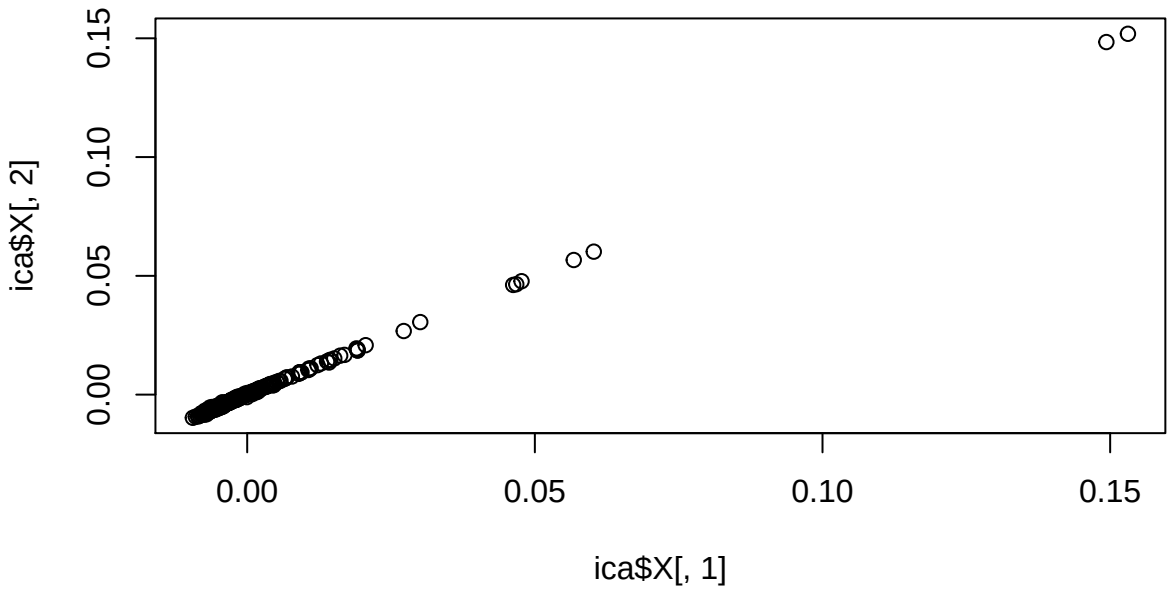


Figure 13: Plot of PC2 over PC1 using ICA

Code appendix

```
#setup
knitr::opts_chunk$set(echo = FALSE, warning=F)
library(MASS)
library(readxl)
library(knitr)
library(tree)
library(e1071)
library(ROCR)
library(fastICA)

#1.1
data1 = read.csv("australian-crabs.csv")

male_data = data1[data1$sex == "Male",]
female_data = data1[data1$sex == "Female",]

plot(data1$CL, data1$RW, ylab="Rear Width", xlab=" Carapace length")
points(male_data$CL, male_data$RW, col="blue")
points(female_data$CL, female_data$RW, col="red")

legend(35, 10, legend=c("Male crabs", "Female crabs"),
       col=c("blue", "red"), lty=1, cex=0.8)
```

```r
#1.2
lda_crabs = lda(sex ~ CL + RW, data=data1, CV=TRUE)
pred_female_data = data1[lda_crabs$posterior[,1] > lda_crabs$posterior[,2],]
pred_male_data = data1[lda_crabs$posterior[,2] > lda_crabs$posterior[,1],]

plot(data1$CL, data1$RW, ylab="Rear Width", xlab=" Carapace length")
points(pred_male_data$CL, pred_male_data$RW, col="blue")
points(pred_female_data$CL, pred_female_data$RW, col="red")

legend(35, 10, legend=c("Male crabs", "Female crabs"),
       col=c("blue", "red"), lty=1, cex=0.8)

pred_sex_list = lda_crabs$posterior[,2] > lda_crabs$posterior[,1] # True is male
sex_list = data1$sex == "Male" # True is male

wrongs = length(data1[sex_list != pred_sex_list, ][,1])

missclassification = wrongs / length(data1[,1])

missclassification

#1.3
lda_crabs = lda(sex ~ CL + RW, data=data1, CV=TRUE, prior=c(0.1, 0.9))
pred_female_data = data1[lda_crabs$posterior[,1] > lda_crabs$posterior[,2],]
pred_male_data = data1[lda_crabs$posterior[,2] > lda_crabs$posterior[,1],]

plot(data1$CL, data1$RW, ylab="Rear Width", xlab=" Carapace length")
points(pred_male_data$CL, pred_male_data$RW, col="blue")
points(pred_female_data$CL, pred_female_data$RW, col="red")

legend(35, 10, legend=c("Male crabs", "Female crabs"),
       col=c("blue", "red"), lty=1, cex=0.8)

pred_sex_list = lda_crabs$posterior[,2] > lda_crabs$posterior[,1] # True is male
sex_list = data1$sex == "Male" # True is male

wrongs = length(data1[sex_list != pred_sex_list, ][,1])

missclassification = wrongs / length(data1[,1])

missclassification

#1.4
crab_glm = glm(sex ~ CL + RW, data = data1, family=binomial())
glm_pred = predict(crab_glm, type="response")

male_glm = data1[glm_pred >= 0.5,]
female_glm = data1[glm_pred < 0.5,]

plot(data1$CL, data1$RW, ylab="Rear Width", xlab=" Carapace length")
points(male_glm$CL, male_glm$RW, col="blue")
points(female_glm$CL, female_glm$RW, col="red")
```

```
slope = coef(crab_glm)[2]/(-coef(crab_glm)[3])
intercept = coef(crab_glm)[1]/(-coef(crab_glm)[3])
abline(a=intercept, b=slope, col="green")

legend(33, 10, legend=c("Male crabs", "Female crabs", "Decision boundary"),
       col=c("blue", "red", "green"), lty=1, cex=0.8)

glm_sex_list = glm_pred > 0.5 # True is male
sex_list = data1$sex == "Male" # True is male

wrongs = length(data1[sex_list != glm_sex_list, ][,1])

missclassification = wrongs / length(data1[,1])

#missclassification
#slope
#intercept

#2.1
data2 = read_excel("creditscoring.xls")
n=dim(data2)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train2=data2[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid2=data2[id2,]

id3=setdiff(id1,id2)
test2=data2[id3,]

#2.2 Deviance
dev_tree = tree(as.factor(good_bad)~., data=train2, split="deviance")
dev_pred_train = predict(dev_tree, newdata=train2, type="class")
dev_pred_test = predict(dev_tree, newdata=test2, type="class")

miss_pred_train = sum(dev_pred_train != train2$good_bad) / nrow(train2)
#miss_pred_train

miss_pred_test = sum(dev_pred_test != test2$good_bad) / nrow(test2)
#miss_pred_test

#2.2 Gini index
gini_tree = tree(as.factor(good_bad)~., data=train2, split="gini")
gini_pred_train = predict(gini_tree, newdata=train2, type="class")
gini_pred_test = predict(gini_tree, newdata=test2, type="class")

miss_pred_train = sum(gini_pred_train != train2$good_bad) / nrow(train2)
#miss_pred_train

miss_pred_test = sum(gini_pred_test != test2$good_bad) / nrow(test2)
```

```
#miss_pred_test

#2.3
trainScore=rep(0,15)
valScore=rep(0,15)

for(i in 2:15) {
  prunedTree=prune.tree(dev_tree,best=i)
  pred=predict(prunedTree, newdata=valid2, type="tree")
  trainScore[i]=deviance(prunedTree)
  valScore[i]=deviance(pred)
}

plot(2:15, trainScore[2:15], type="b", col="red", ylim=c(250,600), ylab="Deviance",
     xlab="Depth")
points(2:15, valScore[2:15], type="b", col="blue")

legend(10, 595, legend=c("Train Data", "Validiation data"),
       col=c("red", "blue"), lty=1, cex=0.8)

minTrain = match(min(trainScore[2:15]), trainScore[2:15])
minVal = match(min(valScore[2:15]), valScore[2:15])

# +1 because list is from 2:15
minTrainVal = trainScore[minTrain+1]
minValVal = valScore[minVal+1]

points(minTrain+1, minTrainVal+1, bg="red", col="red", pch=19)
points(minVal+1, minValVal+1, bg="blue", col="blue", pch=19)

bestTree = prune.tree(dev_tree, best=minVal+1)
plot(bestTree, caption="bajs")
text(bestTree, pretty=0)
title("Structure of the optimal decision tree")

predicted = predict(bestTree, newdata=test2, type="class")
kable(table(test2$good_bad, predicted),
      caption = "\\label{table:dtt}Confusion matrix for decision tree on test data")

#2.4
naive = naiveBayes(as.factor(good_bad)~., data=train2)
predictedTrain = predict(naive, newdata=train2, type="class")
kable(table(train2$good_bad, predictedTrain),
      caption = "\\label{table:nbtr}Confusion matrix for naive bayes on training data")

predictedTest = predict(naive, newdata=test2, type="class")
kable(table(test2$good_bad, predictedTest),
      caption = "\\label{table:nbt}Confusion matrix for naive bayes on test data")

#2.5
tree_pred = predict(bestTree, newdata=test2)
bayes_pred = predict(naive, newdata=test2, type = "raw")
```

```
roc_frame = data.frame(
  pi = double(),
  tree_tpr = double(),
  tree_fpr = double(),
  bayes_tpr = double(),
  bayes_fpr = double()
)

for(pi in seq(0.05,0.95,0.05)){
  tree_class = factor(ifelse(tree_pred[,2]>pi, "good", "bad"), levels=c("bad", "good"))
  bayes_class = factor(ifelse(bayes_pred[,2]>pi, "good", "bad"), levels=c("bad", "good"))

  tree_table = table(test2$good_bad, tree_class)
  tree_tpr = tree_table[2,2] / sum(tree_table[2,])
  tree_fpr = tree_table[1,2] / sum(tree_table[1,])

  bayes_table = table(test2$good_bad, bayes_class)
  bayes_tpr = bayes_table[2,2] / sum(bayes_table[2,])
  bayes_fpr = bayes_table[1,2] / sum(bayes_table[1,])

  new_row = data.frame(pi=pi,
                       tree_tpr = tree_tpr,
                       tree_fpr = tree_fpr,
                       bayes_tpr = bayes_tpr,
                       bayes_fpr = bayes_fpr)
  roc_frame = rbind(roc_frame, new_row)
}


plot(roc_frame$tree_fpr, roc_frame$tree_tpr, type="l",ylim=c(0,1), xlim=c(0,1),col="blue",
     xlab="FPR", ylab="TPR")
lines(roc_frame$bayes_fpr, roc_frame$bayes_tpr,col="red")
abline(a=0,b=1)

legend(0.7,0.4, legend=c("Tree", "Bayes"), col=c("blue", "red"), lty=1)

#2.6
loss_matrix = matrix(c(0,1,10,0), nrow=2, ncol=2)

#train
predicted_naive_train = predict(naive, newdata=train2, type="raw")
predicted_lm_train = as.factor(ifelse(predicted_naive_train[,2]/predicted_naive_train[,1]
                                      > loss_matrix[1,2]/loss_matrix[2,1], "good", "bad"))

lm_table_train = table(train2$good_bad, predicted_lm_train)
kable(lm_table_train,
      caption = "\\label{table:lmntr}Confusion matrix for naive bayes on train data using loss matrix")

#test
predicted_naive_test = predict(naive, newdata=test2, type="raw")
predicted_lm_test = as.factor(ifelse(predicted_naive_test[,2]/predicted_naive_test[,1]
                                     > loss_matrix[1,2]/loss_matrix[2,1], "good", "bad"))
```

```
lm_table_test = table(test2$good_bad, predicted_lm_test)
kable(lm_table_test,
      caption = "\\label{table:lmnt}Confusion matrix for naive bayes on test data using loss matrix")

#4.1
data4 = read.csv2("NIRSpectra.csv", sep=";")
data4$Viscosity = c()
res=prcomp(data4)
screeplot(res)

plot(res$x[,1], res$x[,2], xlab="PC1", ylab="PC2")
points(res$x[,1][res$x[,1] > 0.3], res$x[,2][res$x[,1] > 0.3], pch=19, col="red")

U=res$rotation
plot(U[,1], main="Traceplot, PC1")
plot(U[,2], main="Traceplot, PC2")
set.seed(12345)
ica = fastICA(data4, n.com=2)
w_prim = ica$K %*% ica$W
plot(w_prim[,1])
plot(w_prim[,2])
plot(ica$X[,1], ica$X[,2])
```