TDDE01 Machine Learning

Lab2

Ruben Hillborg

November 30, 2018

Assignment 1. LDA and logistic regression

A file containing measurements of 200 different Austrailian crabs were given. The different measurements were frontal lobe, rear width, carapace length and others. The sex of the crabs were also available.

1.1 Scatterplot

A scatterplot of carapace length vs rear width where the data points were colored according to the sex of the crabs was created. The scatterplot can be seen in figure 1. From the plot it looks like the two sexes each could be described by a different linear model. Some overlapping datapoints for lower values of both CL and RW can also be observed. Because of this, the data sould be pretty easy to classify using LDA on large values of CL and RW, but there should be some missclassifications for lower values.

1.2 Proporional LDA

LDA classification was used on the same features as in section 1.1. The resulting predictions can be seen in figure 2. The classifier does a good job with a **missclassification error** of only 0.035. As predicted in section 1.1, most missclassifications happens for the lower values of CL and RW, where the sexes overlaps much.

1.3 Unproportional LDA

LDA classification was used once again, but this time with a prior of $p(Male) = 0.9$, $p(Female) = 0.1$. The result was a worse classifier than in section 1.2, as can be seen in figure 3. The **missclassification error** went up to 0.08. From the plot we can see that the high prior probability of a crab being male has made the classifier predic almost all of the overlapping data points as male.

1.4 Logistic regression

Lastly logistic regression was used to try and classify the data. The results can be seen in figure 4. The plot looks a lot like the plot in section 2.1. The **missclassification error** was 0.035, which is exactly the same as in section 2.1.

Asssignment 2. Analysis of credit scoring

In this assignment a file containing data about customers from a private enterprise was given. The data consisted of a variable indicating if the customer had managed their loans good or bad. The other features should be used to derive a prediction model that could be used to predic whether or not a new customer were likely to pay back their loan.

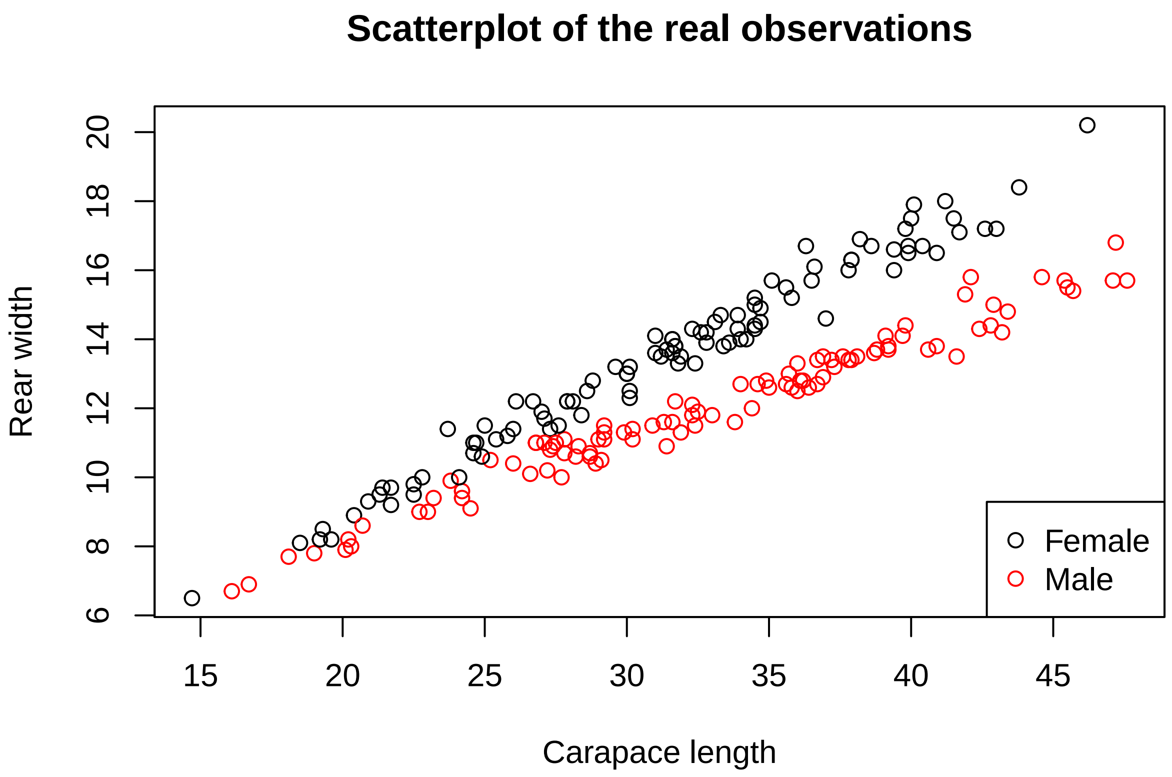


Figure 1: Scatterplot of carapace length vs rear width, colored by sex

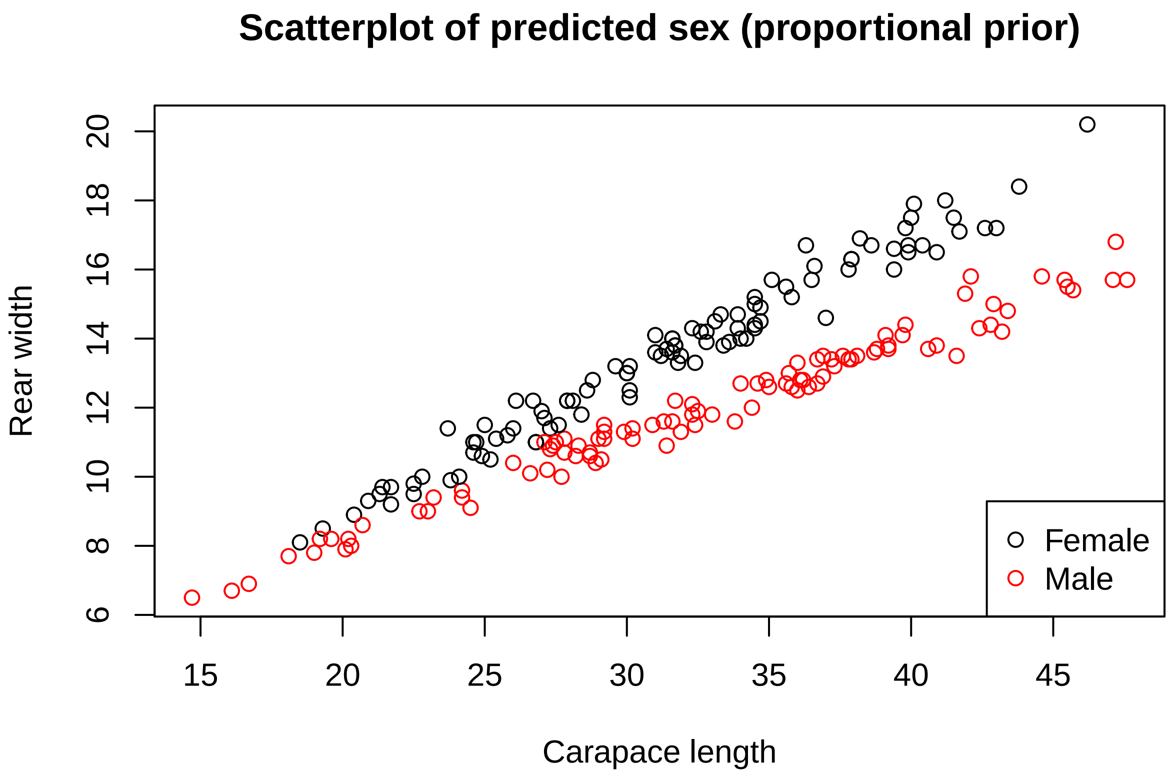


Figure 2: Scatterplot of carapace length vs rear width, colored by predicted sex

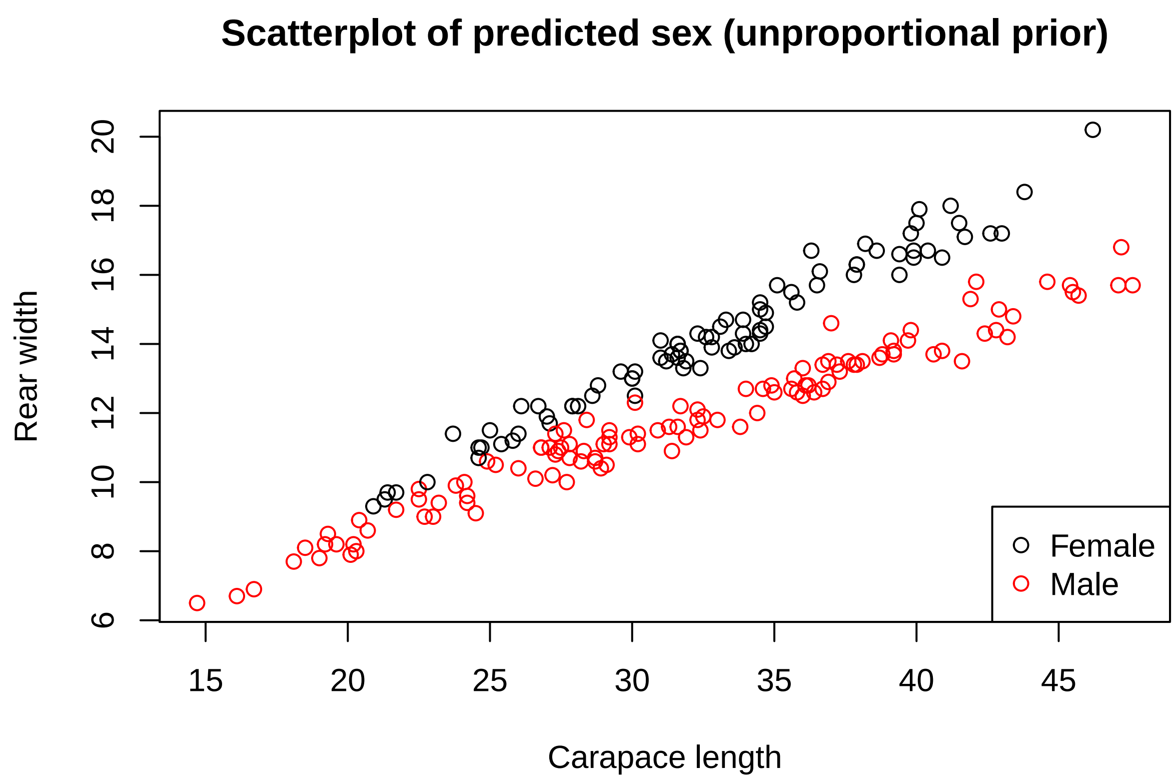


Figure 3: Scatterplot of carapace length vs rear width, colored by predicted sex

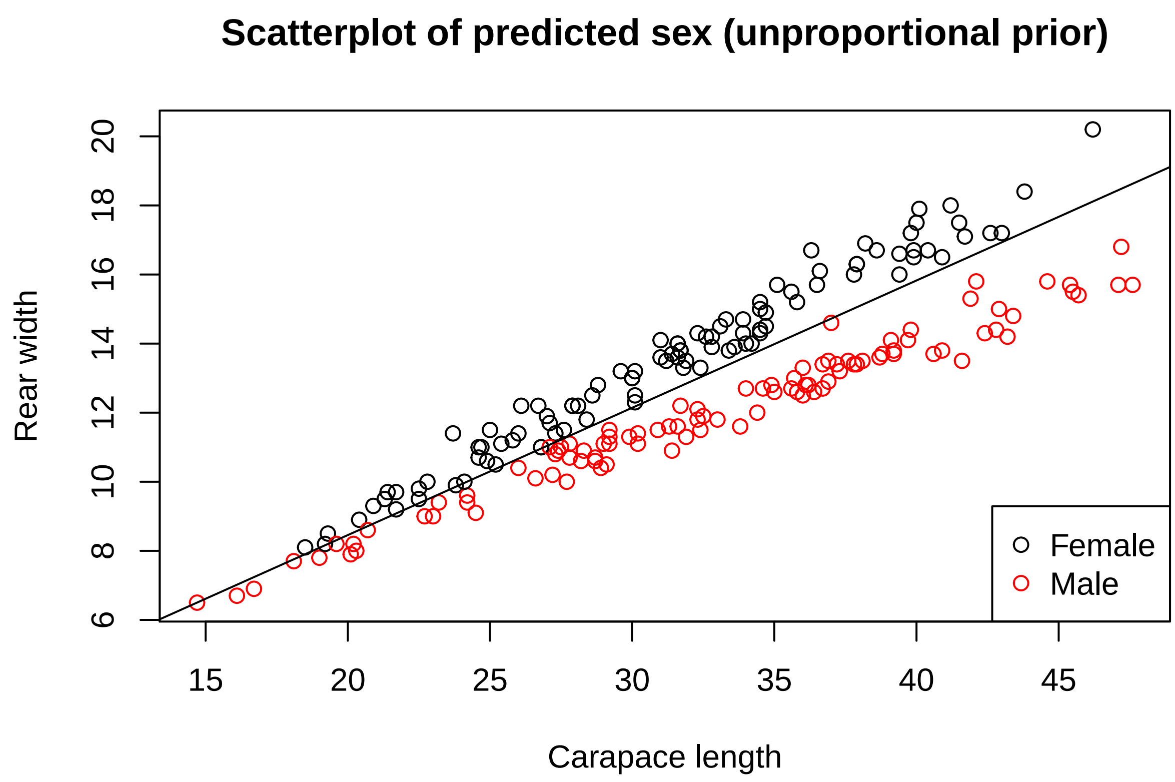


Figure 4: Scatterplot of carapace length vs rear width, colored by predicted sex

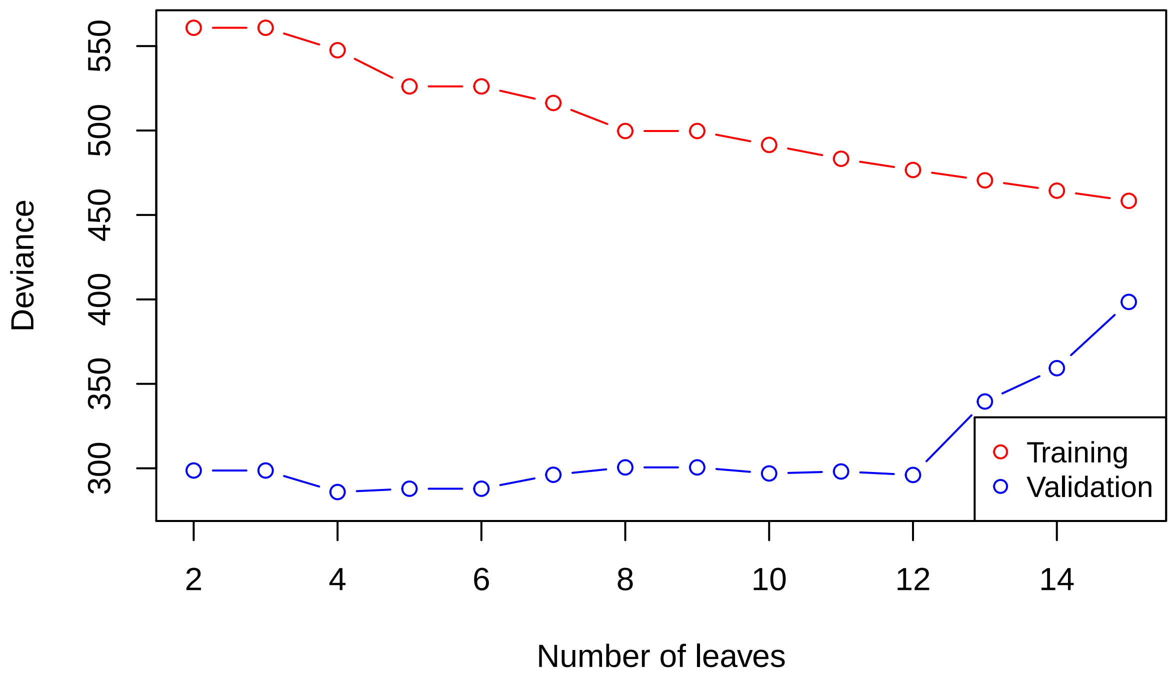


Figure 5: Number of leaves vs deviance for training and validation data.

2.1 Setup

The data was first divided into three datasets: training/validation/test as 50/25/25.

2.2 Decision tree

Two decision trees were fitted to the training data, one using deviance and the other using gini index as the measure of impurity. The resulting missclassification errors were:

Missclassification error, deviance, training set: 0.212
Missclassification error, deviance, testing set: 0.268
Missclassification error, gini index, training set: 0.236
Missclassification error, gini index, testing set: 0.36

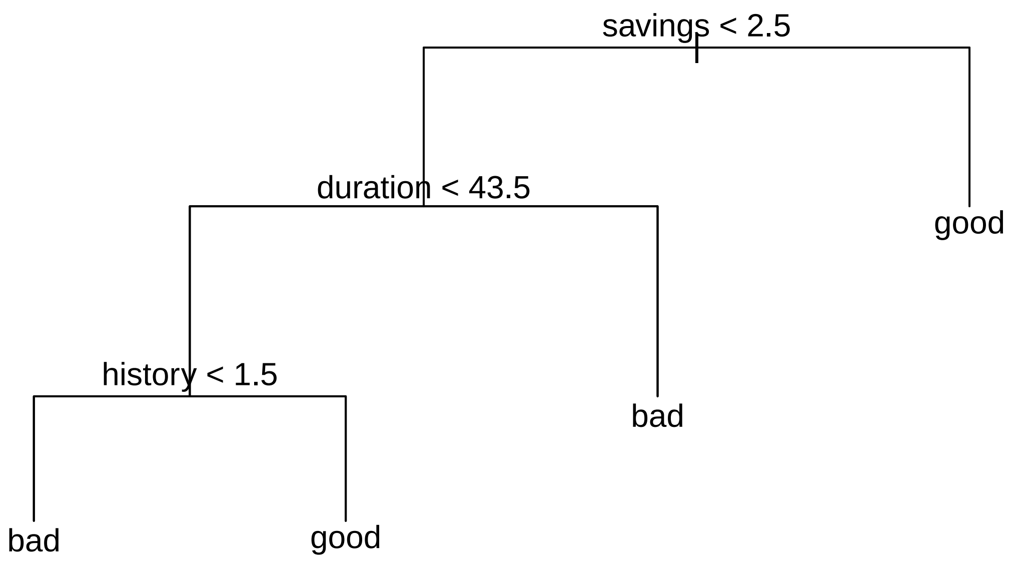
We can see that the tree fitted using deviance has a lower missclassification error on both datasets, so the following assignments were completed using this tree.

2.3 Finding the optimal tree

Table 1: Confusion matrix for the optimal tree used on validation data

	Pred bad	Pred good
bad	23	54
good	12	161

The training and validation data was used to prune the tree for all different numbers of leaves possible. The results can be seen in figure 5. From the plot we can see that the optimal tree consists of the tree pruned to only 4 leaves. The optimal tree can be seen below:



We can see that the optimal tree has a depth of 3, and uses the variables savings, duration and history. The tree tells us that the amount of savings a person has contributes the most towards how the person will manage their loan. The next most important factor is the duration of the loan, and the third most important factor is the persons previous history of loans. The **missclassification rate** for the optimal tree predicting the test data was: 0.264, and the confusion matrix can be seen in table 1.

2.4 Naive Bayes

Table 2: Confusion matrix for Naive Bayes used on training data

	Pred bad	Pred good
bad	95	52
good	98	255

Table 3: Confusion matrix for Naive Bayes used on test data

	Pred bad	Pred good
bad	46	30
good	49	125

To compare the results of the optimal tree found in section 2.3, Naive Bayes classification was used on the training and testing data. The resulting confusion matrices can be seen in tables 2 and 3. The missclassification rates were:

Missclassification rate for training set: 0.3
Missclassification rate for testing set: 0.316

Both missclassification rates were higher than for the missclassification rates for the optimal tree found in section 2.3. The tree is also easy to interpret just by looking at the plot of the tree, which could be an advantage too.

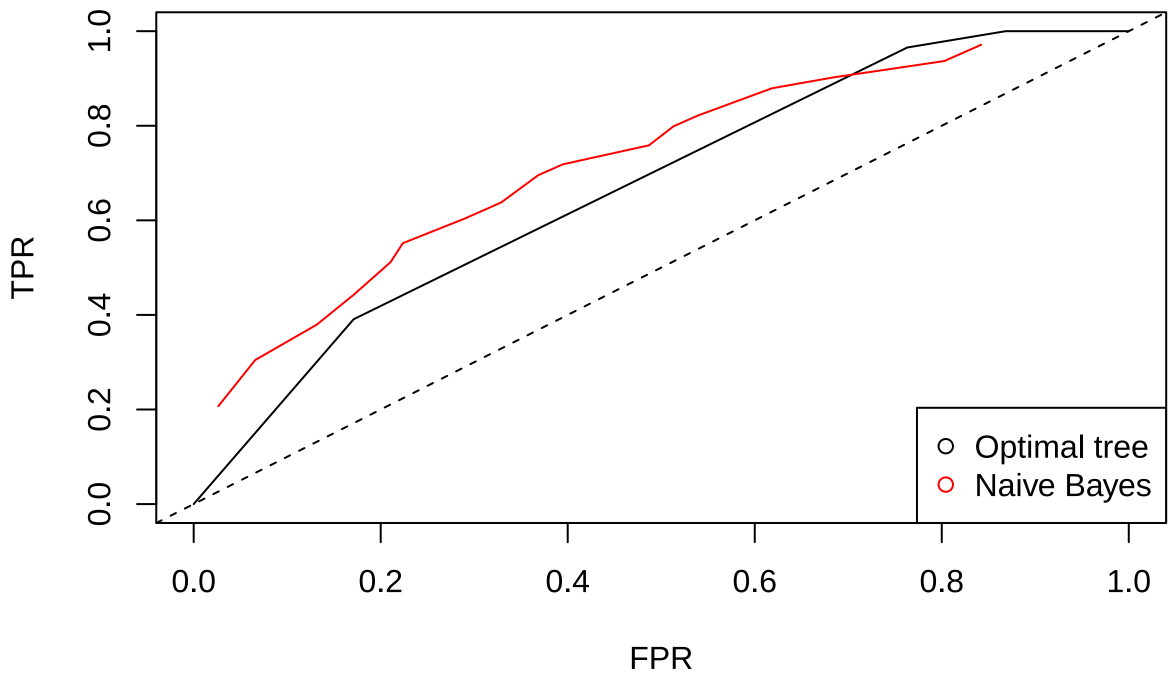


Figure 6: The ROC curves for the optimal tree and Naive Bayes used on the test data.

2.5 ROC curves

Both the optimal tree and Naive Bayes were used to classify the test data by using the following classification principle:

$$\hat{Y} = 1 \; if \; p(Y = good|X) > \pi, \; otherwise \; \hat{Y} = 0$$

where $\pi = 0.05, 0.1, 0.15, \ldots, 0.9, 0.95$. The resulting ROC curves can be found in figure 6. In the plot we can see that the area under the curve (AUC) for Naive Bayes is larger than that of the optimal tree, which is probably wrong since the optimal tree had a better missclassification rate than Naive Bayes had.

2.6 Naive Bayes with loss matrix

```
##         train_res
##          bad good
##    bad   137   10
##    good 263   90
```

```
##         test_res
##          bad good
##    bad    71    5
##    good 122   52
```

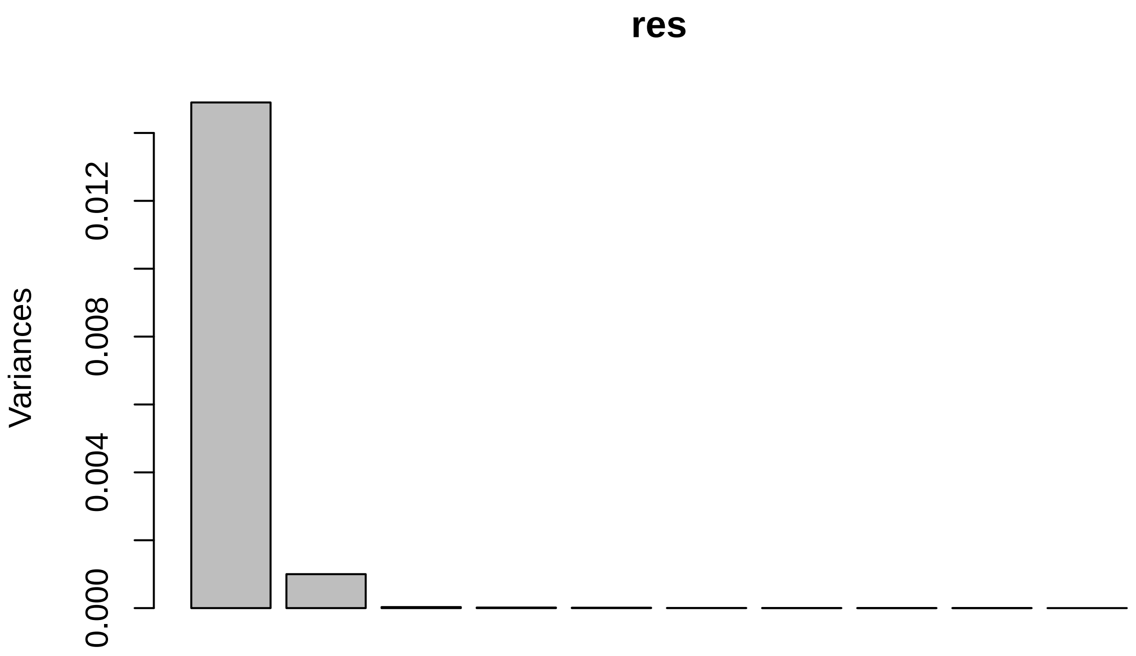
Lastly Naive Bayes was used again, but this time together with a loss matrix. This made the prediction a lot worse.

Assignment 4. Principal components

4.1

PCA was used on the data, and the results can be seen in the plot. We can see that only 2 of the components explains over 99% of the total variance of the data. We can also see that there are some outliers in the second plot, telling us that there are some unusual diesel fuels in the data.

```
##   [1] "93.332" "6.263"  "0.185"  "0.101"  "0.068"  "0.025"  "0.009"
##   [8] "0.003"  "0.003"  "0.002"  "0.001"  "0.001"  "0.001"  "0.001"
##  [15] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [22] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [29] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [36] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [43] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [50] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [57] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [64] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [71] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [78] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [85] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [92] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [99] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [106] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [113] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [120] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
```



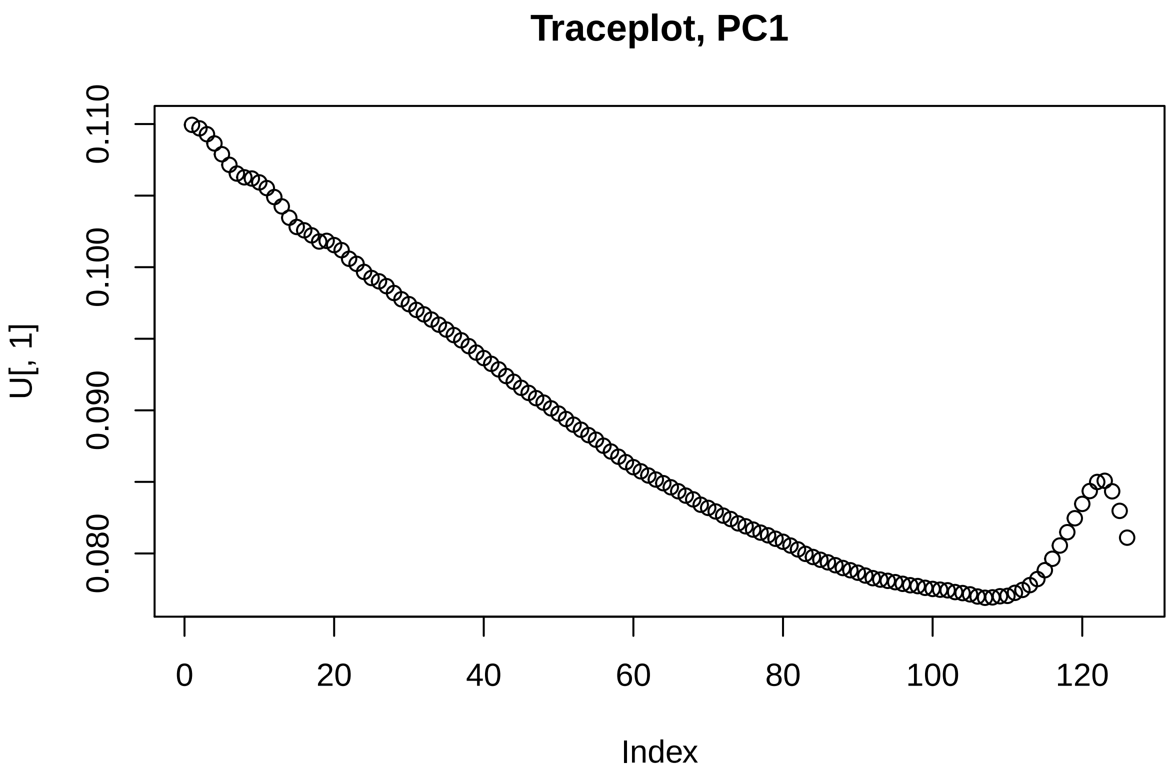
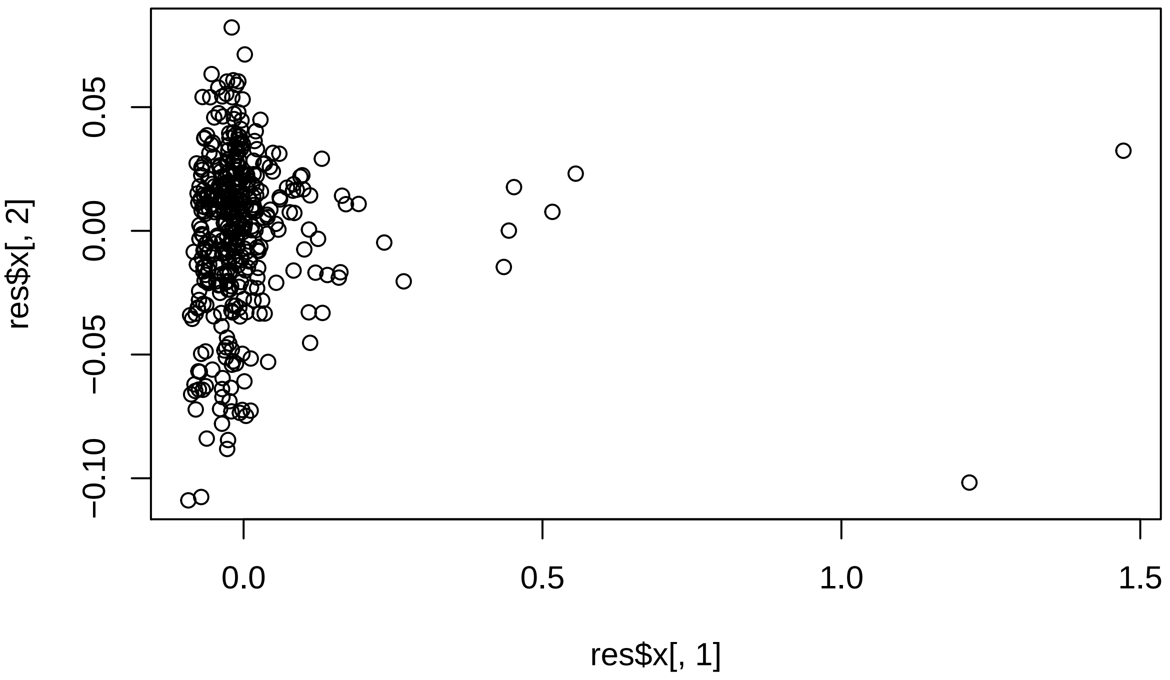


Figure 7: Traceplot of PC1



4.2

Two traceplots, one for PC1 and one for PC2 were created. These can be seen in figures 7 and 8. All points in figure 7 have similar values (look at the scale of the y-axis), while the peak values in figure 8 are much

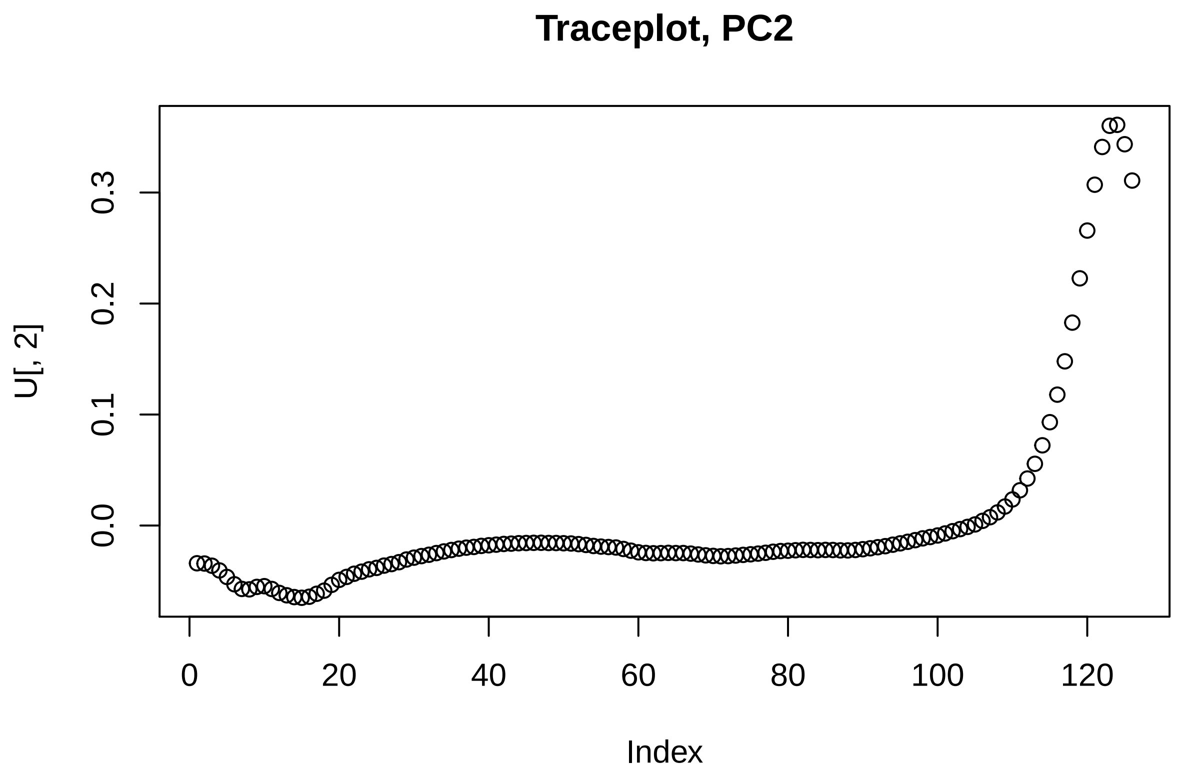
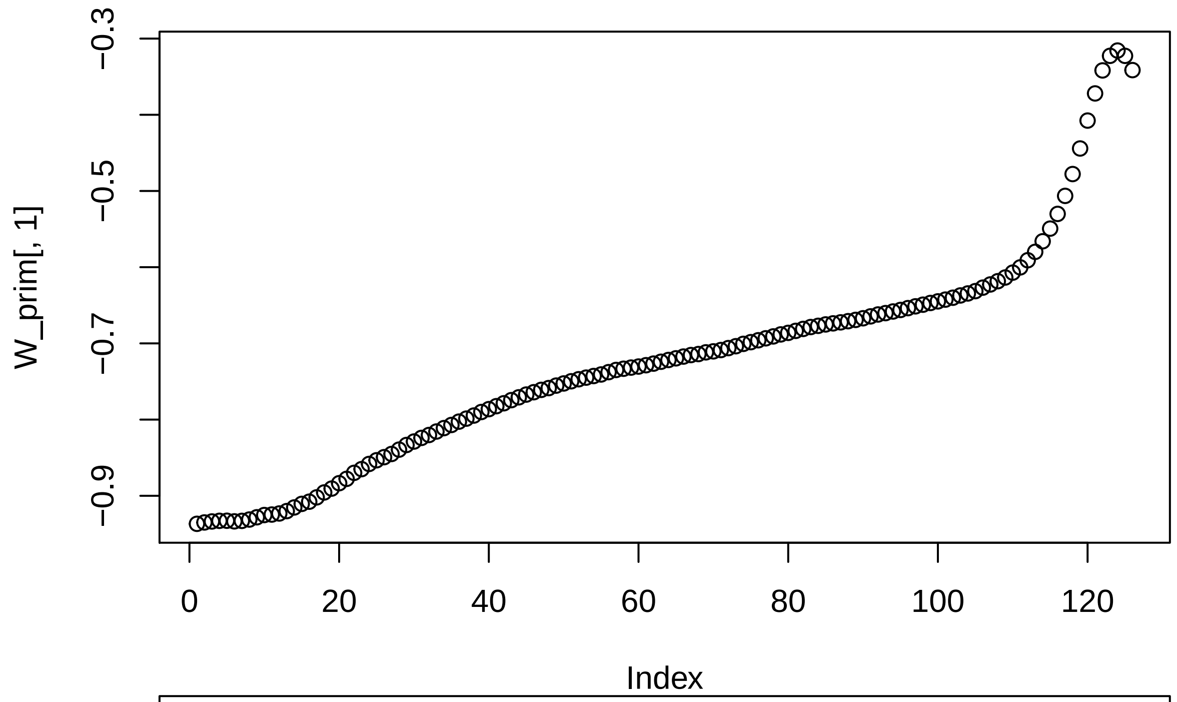


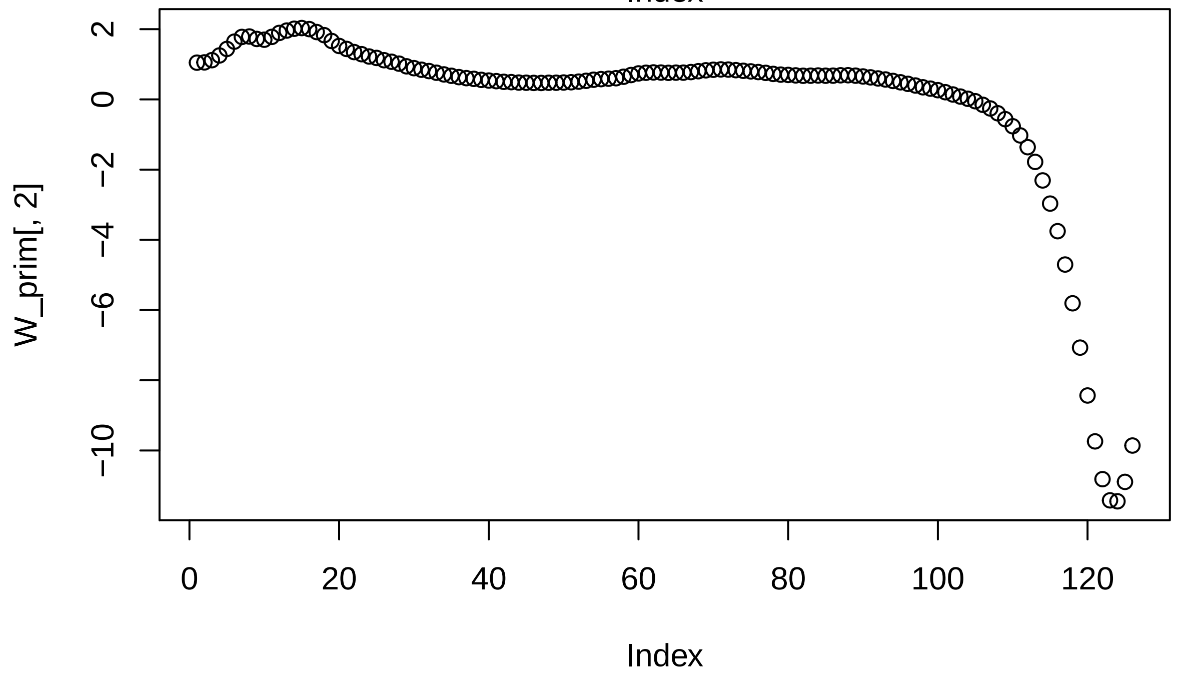
Figure 8: Traceplot of PC2

larger than in the rest of the plot. Because of this it looks like PC2 is based mainly on just a few of the original features, while PC1 isn't.

4.3

The matrix W' contains the measures of the ICA loadings, and can be found plotted below:





Code appendix

```
knitr::opts_chunk$set(echo = FALSE)
require(MASS)
library(readxl)
library(tree)
library(e1071)
library(knitr)
```

```r
library(fastICA)

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

plot(data1$CL, data1$RW,
     col = data1$sex,
     main = "Scatterplot of the real observations",
     xlab = "Carapace length",
     ylab = "Rear width")
legend("bottomright", legend = levels(data1$sex), col = 1:2, pch = 1)

# 1.2
lda = lda(sex ~ CL + RW, data = data1)
lda.values = predict(lda, newdata = data1)
lda.mcr = mean(data1$sex != lda.values$class)
plot(data1$CL, data1$RW,
     col = lda.values$class,
     main = "Scatterplot of predicted sex (proportional prior)",
     xlab = "Carapace length",
     ylab = "Rear width")
legend("bottomright", legend = levels(data1$sex), col = 1:2, pch = 1)

# 1.3
lda2 = lda(sex ~ CL + RW, data = data1, prior = c(0.1, 0.9))
lda2.values = predict(lda2, newdata = data1)
lda2.mcr = mean(data1$sex != lda2.values$class)
plot(data1$CL, data1$RW,
     col = lda2.values$class,
     main = "Scatterplot of predicted sex (unproportional prior)",
     xlab = "Carapace length",
     ylab = "Rear width")
legend("bottomright", legend = levels(data1$sex), col = 1:2, pch = 1)

# 1.4
glm = glm(sex ~ CL + RW, data = data1, family = "binomial")
glm.predictions = ifelse(predict(glm, type = "response") > 0.5, 2, 1)
glm.mcr = mean(as.integer(data1$sex) != glm.predictions)
plot(data1$CL, data1$RW,
     col = glm.predictions,
     main = "Scatterplot of predicted sex (unproportional prior)",
     xlab = "Carapace length",
     ylab = "Rear width")
legend("bottomright", legend = levels(data1$sex), col = 1:2, pch = 1)
intercept = coef(glm)[1]/(-coef(glm)[3])
slope = coef(glm)[2]/(-coef(glm)[3])
abline(intercept, slope)

# 2.1
data2 = read_excel("creditscoring.xls")

n=dim(data2)[1]
set.seed(12345)
```

```r
id=sample(1:n, floor(n*0.5))
train=data2[id,]

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

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

#2.2
set.seed(12345)
tree_dev = tree(as.factor(good_bad) ~ ., data = train, split = "deviance")
tree_dev_train_pred = predict(tree_dev, newdata = train, type = "class")
tree_dev_train_mcr = mean(train$good_bad != tree_dev_train_pred)
tree_dev_test_pred = predict(tree_dev, newdata = test, type = "class")
tree_dev_test_mcr = mean(test$good_bad != tree_dev_test_pred)

tree_gini = tree(as.factor(good_bad) ~ ., data = train, split = "gini")
tree_gini_train_pred = predict(tree_gini, newdata = train, type = "class")
tree_gini_train_mcr = mean(train$good_bad != tree_gini_train_pred)
tree_gini_test_pred = predict(tree_gini, newdata = test, type = "class")
tree_gini_test_mcr = mean(test$good_bad != tree_gini_test_pred)

# 2.3
set.seed(12345)
leafs = 0:15
train_score = rep(0, 15)
test_score = rep(0, 15)

for(i in 2:15) {
  pruned_tree = prune.tree(tree_dev, best=i)
  pred = predict(pruned_tree, newdata = valid, type = "tree")
  train_score[i] = deviance(pruned_tree)
  test_score[i] = deviance(pred)
}

plot(2:15, train_score[2:15],
     type = "b",
     col = "red",
     xlab = "Number of leaves",
     ylab = "Deviance",
     ylim = c(280, 560))
points(2:15, test_score[2:15], type = "b", col = "blue")
legend("bottomright", legend = c("Training", "Validation"), col = c("red", "blue"), cex = 0.9, pch = 1)

optimal_tree = prune.tree(tree_dev, best=4)
ot_pred = predict(optimal_tree, newdata = valid, type="class")
kable(table(valid$good_bad, ot_pred), caption = "\\label{table:ot_cm} Confusion matrix for the optimal
mcr = mean(valid$good_bad != ot_pred)

plot(optimal_tree)
```

```r
text(optimal_tree)

# 2.4
nb = naiveBayes(as.factor(good_bad) ~ ., data = train)
nb_train_pred = predict(nb, newdata = train)
kable(table(train$good_bad, nb_train_pred), caption = "\\label{table:nb_train_cm} Confusion matrix for N
nb_train_mcr = mean(nb_train_pred != train$good_bad)

nb_test_pred = predict(nb, newdata = test)
kable(table(test$good_bad, nb_test_pred), caption = "\\label{table:nb_test_cm} Confusion matrix for Nai
nb_test_mcr = mean(nb_test_pred != test$good_bad)

# 2.5
roc = data.frame(ot_tpr=double(), ot_fpr=double(), nb_tpr=double(), nb_ftr=double())

ot_pred = predict(optimal_tree, newdata = test)
nb_pred = predict(nb, newdata = test, type = "raw")

for (pi in seq(0.05, 0.95, 0.05)) {
  predictions = as.factor(ifelse(ot_pred[,2] > pi, "good", "bad"))

  ot_cm = matrix(0, nrow = 2, ncol = 2)

  ot_cm[1,1] = sum(predictions == "bad" & as.factor(test$good_bad) == "bad")
  ot_cm[1,2] = sum(predictions == "good" & as.factor(test$good_bad) == "bad")
  ot_cm[2,1] = sum(predictions == "bad" & as.factor(test$good_bad) == "good")
  ot_cm[2,2] = sum(predictions == "good" & as.factor(test$good_bad) == "good")

  ot_tpr = as.double(ot_cm[2,2]) / as.double(sum(ot_cm[2,]))
  ot_fpr = as.double(ot_cm[1,2]) / as.double(sum(ot_cm[1,]))

  predictions = as.factor(ifelse(nb_pred[,2] > pi, "good", "bad"))
  nb_cm = matrix(0, nrow = 2, ncol = 2)

  nb_cm[1,1] = sum(predictions == "bad" & test$good_bad == "bad")
  nb_cm[1,2] = sum(predictions == "good" & test$good_bad == "bad")
  nb_cm[2,1] = sum(predictions == "bad" & test$good_bad == "good")
  nb_cm[2,2] = sum(predictions == "good" & test$good_bad == "good")

  nb_tpr = as.double(nb_cm[2,2]) / as.double(sum(nb_cm[2,]))
  nb_fpr = as.double(nb_cm[1,2]) / as.double(sum(nb_cm[1,]))

  row = data.frame(ot_tpr = ot_tpr, ot_fpr = ot_fpr, nb_tpr = nb_tpr, nb_fpr = nb_fpr)
  roc = rbind(roc, row)
}

plot(roc$ot_fpr, roc$ot_tpr,
     xlab = "FPR",
     ylab = "TPR",
     type = "l")
points(roc$nb_fpr, roc$nb_tpr,  type = "l", col = "red")
legend("bottomright", legend = c("Optimal tree", "Naive Bayes"), col = 1:2, pch = 1)
abline(0, 1, lty = 2)
```

```r
# 2.6
lm = matrix(0, nrow = 2, ncol = 2)
lm[1,1] = 0
lm[1,2] = 1
lm[2,1] = 10
lm[2,2] = 0

nb = naiveBayes(as.factor(good_bad) ~ ., data = train)

nb_train_pred = predict(nb, newdata = train, type = "raw")
train_res = ifelse(nb_train_pred[,2] / nb_train_pred[,1] > lm[2,1]/lm[1,2], "good", "bad")
table(train$good_bad, train_res)

nb_test_pred = predict(nb, newdata = test, type = "raw")
test_res = ifelse(nb_test_pred[,2] / nb_test_pred[,1] > lm[2,1]/lm[1,2], "good", "bad")
table(test$good_bad, test_res)

# 4.1
data3 = read.csv2("NIRSpectra.csv")
data3$Viscosity = c()
res=prcomp(data3)
lambda=res$sdev^2
#proportion of variation
sprintf("%2.3f",lambda/sum(lambda)*100)
screeplot(res)
plot(res$x[,1], res$x[,2])

# 4.2
U=res$rotation
plot(U[,1], main="Traceplot, PC1")
plot(U[,2],main="Traceplot, PC2")

# 4.3
set.seed(12345)
ica = fastICA(data3, 2)
W_prim = ica$K %*% ica$W
plot(W_prim[,1])
plot(W_prim[,2])
```