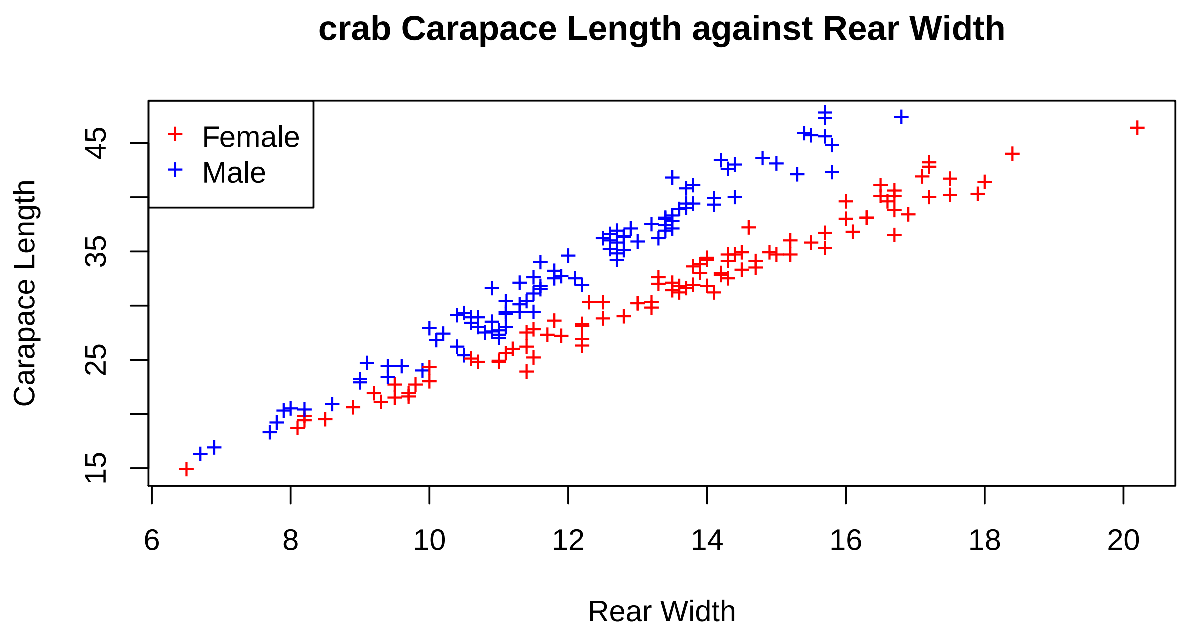
Computer Lab 3

Thomas Zhang

2015-11-09

Assignment 1

For the data in file `australian-crabs.csv` we make a scatter plot of crab carapace length (CL) against Rear Width (RW).



We can clearly distinguish two separate linear trends in the scatterplot. One trend for the male crabs and one trend for the female crabs. I think it will be easy to classify the sex for the larger crabs, since at larger values of CL and RW they are well separated from each other. At smaller values for the two variables it will be harder to classify with linear discriminant analysis (LDA), since the two trends are not well separated there.

The discriminant function used in LDA can be written as:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log(\pi_k)$$

where the estimates for Σ, μ_k and π_k are given, with c denoting class k, N denoting the total number of crabs and N_c the total number of crabs in class c, as

$$\hat{\mu}_c = \frac{1}{N_c}\Sigma_{i:y_i=c}x_i$$

$$\hat{\Sigma}_c = \frac{1}{N_c}\Sigma(x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T$$

$$\hat{\Sigma} = \frac{1}{N}\Sigma_{c=1}^{k} N_c \hat{\Sigma}_c$$

$$\hat{\pi}_c = \frac{N_c}{N}$$

After fitting the model, the discriminant function for the male crabs $\delta_{male}(x)$ can be written:

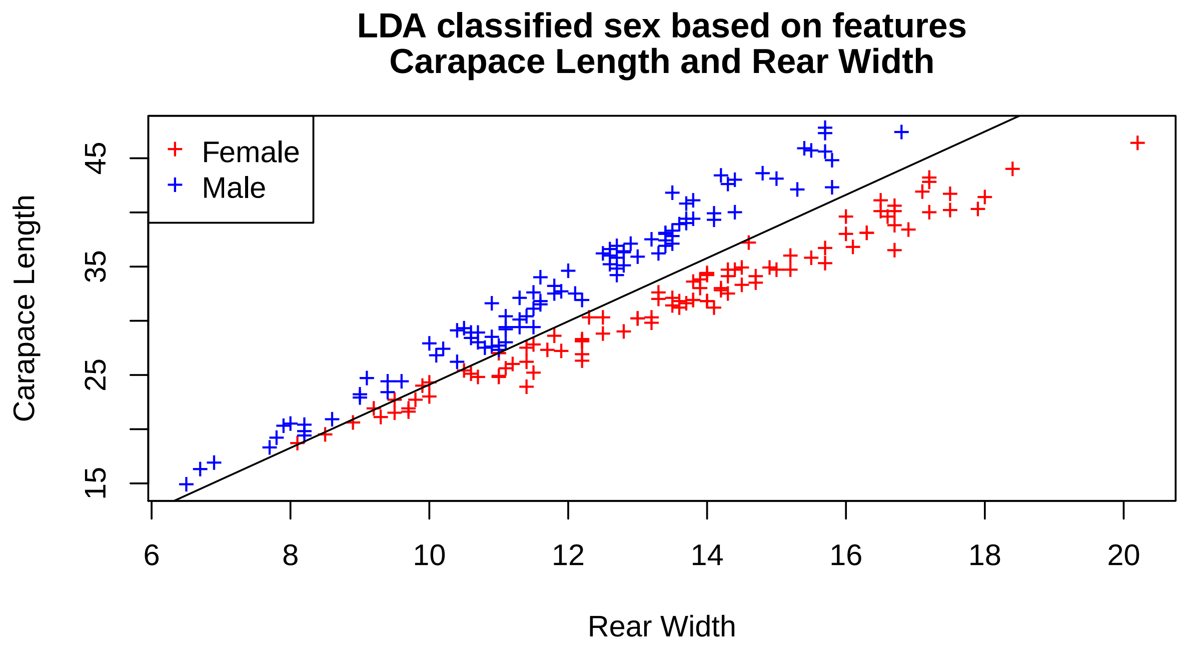
$$\delta_{male}(x) = x^T \begin{pmatrix} 2.592 \\ -0.216 \end{pmatrix} - 12.683$$

and the discriminant function for the female crabs $\delta_{female}(x)$ can be written:

$$\delta_{female}(x) = x^T \begin{pmatrix} 8.332 \\ -2.183 \end{pmatrix} - 22.648$$

The decision boundary is drawn where $\delta_{male}(x) = \delta_{female}(x)$.

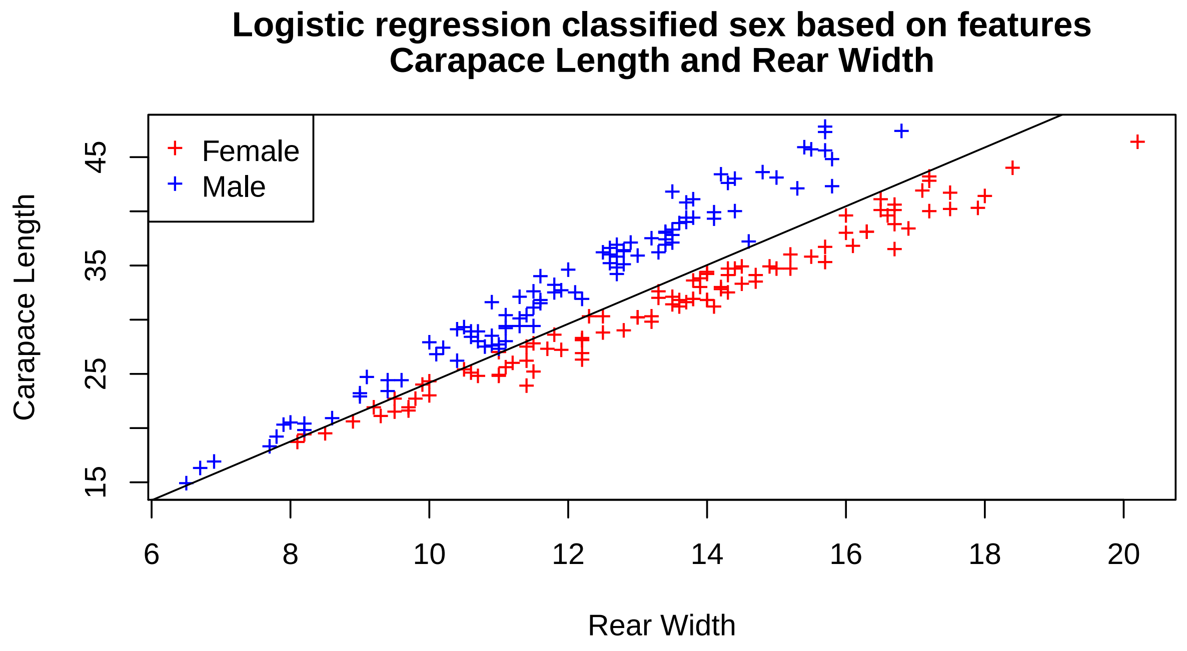
Proceeding now to perform home-made LDA and plot the results:



```
## [1] "Decision boundary: intercept =  -5.066 , slope=  2.918"
```

It seems we get rather good results, comparing with the truth only a few of the smaller crabs are misclassified by sex.

Now we classify the crabs by sex using the method of logistic regression. We use the R method `glm`.



```
## [1] "Decision boundary: intercept =  -2.94 , slope=  2.713"
```

We see that the slope is a little bit less steep in the logistic regression decision boundary as compared to the LDA decision boundary. The performance is about the same as LDA though.

Assignment 2

We import data from file `creditscoring.xls`. The data frame contains information about 1000 borrowers and their loans. We wish to fit decision trees for classification of variable `good_bad`, a variable describing whether the person is a good or a bad credit risk. We divide the data into training, validation and test data sets using a 50/25/25 split and start fitting two decision trees using the training data set. We use two measures of node impurity. For `fittree1` we use `split = "deviance"` which essentialy minimizes tree information entropy to pick the tree splits, and for `fittree2` we use `split = "gini"`, which minimizes tree Gini impurity. We find misclassification rates for the test data for the two trees.

```
## [1] "Misclassification rate of fittree1 for training data:  0.21"
```

```
## [1] "Misclassification rate of fittree1 for test data:  0.292"
```

```
## [1] "Misclassification rate of fittree2 for training data:  0.252"
```

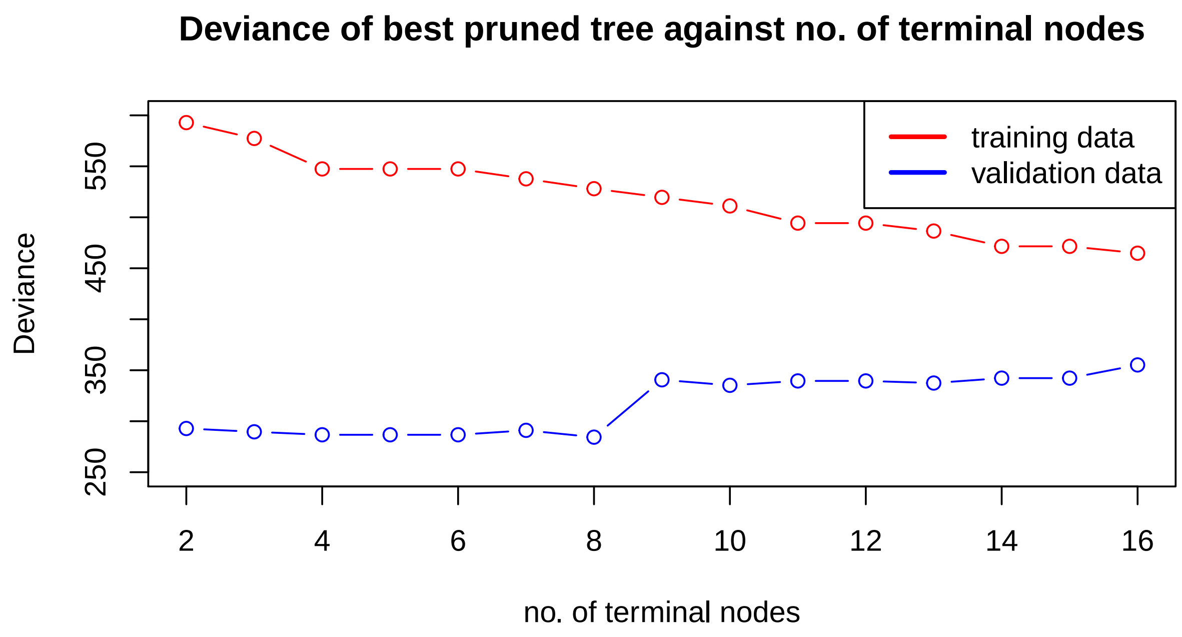
```
## [1] "Misclassification rate of fittree2 for test data:  0.296"
```

We see that the impurity measure of information entropy is better, so we will continue to evaluate the decision tree `fittree1`.

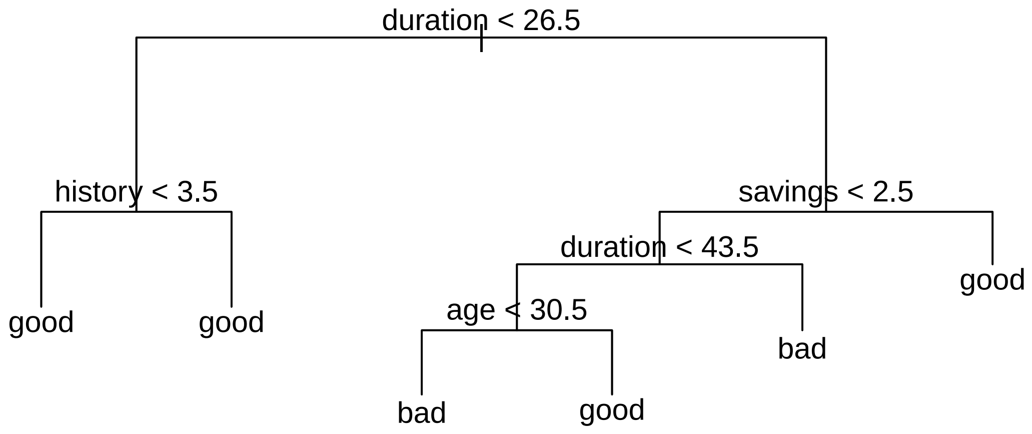
We are going to prune the tree `fittree1` by minimizing

$$\text{Deviance of tree} + \lambda|T|$$

where $|T|$ is the number of terminal nodes and λ is an algorithmically determined cost parameter. Below you will find the plot of deviance of the tree for training data and for validation data against number of terminal nodes in the greedily and recursively pruned subtree of `fittree1`.



It looks like the deviance of tree for validation data minimizes around tree size (size meaning no. of terminal nodes in tree) eight, but after inspection of the tree we realize that just because the mathematical definition of deviance is minimized, it does not prevent the tree from being unnecessarily large. In reality, the best pruned tree of size five classifies exactly the same way (the two left-most terminal nodes can be considered one node).



```
##       predtestbest
##        bad good
##   bad    9   56
##   good  12  173
```

```
## [1] "Misclassification rate of besttree for test data:  0.272"
```

We see that this pruned tree has a better misclassification rate for test data than the unpruned tree `fittree1`.

From the nodes of this tree, it seems having long duration loans, low levels of savings and being young makes one a “bad” credit risk. according to this tree (which it must be said is quite simplistic a view, since many “bad” are misclassified as “good” in this tree)

Lastly we will fit a naive Bayes classifier model to the training data, and find the misclassification rates for training data and test data.

```
##       nbayespredtrain
##        bad good
##   bad  116   47
##   good 115  222
```

```
##       nbayespredtest
##        bad good
##   bad   46   19
##   good  76  109
```

```
## [1] "Misclassification rate of nbayesmodel for training data:  0.324"
```

```
## [1] "Misclassification rate of nbayesmodel for test data:  0.38"
```

The misclassification rates appear to be higher than for the unpruned decision trees. We now use a loss matrix L,

$$L = \begin{array}{c c} & \begin{array}{cc} \textit{good} & \textit{bad} \end{array} \\ \begin{array}{c} \textit{good} \\ \textit{bad} \end{array} & \begin{pmatrix} 0 & 1 \\ 10 & 0 \end{pmatrix} \end{array}$$

where the rows are truth values and the columns are classified values in the naive bayes classifier, train a model based on training data and find the confusion matrices generated for the training data and test data.

```
##       preds
##        bad good
##   bad  156    7
##   good 252   85
```

```
## [1] "Misclassification rate of nbayesmodel with loss matrix for training data:  0.518"
```

```
##       predtests
##        bad good
##   bad   63    2
##   good 151   34
```

```
## [1] "Misclassification rate of nbayesmodel with loss matrix for test data:  0.636"
```

we see that there are a lot more in truth “good” credit risks classified as “bad” credit risk and very few vice versa. This happens because we use the loss matrix misclassification penalties which says that a “bad” credit risk classified as “good” is ten times more costly than vice versa, so the naive bayes classifier will produce results based on these penalties. The misclassification rates are very high for both training data and test data but the loss is probably minimized.

Appendix

R code

```r
crabdata <- read.csv("data/australian-crabs.csv")

n <- dim(crabdata)[1]
plot(crabdata$RW,crabdata$CL,pch="+",
     col=ifelse(as.numeric(crabdata$sex) == 1, "red","blue"),
     main="crab Carapace Length against Rear Width",ylab ="Carapace Length",
     xlab="Rear Width")
legend("topleft",legend = levels(crabdata$sex),
                              pch = "+",col=c("red","blue"))


#Yes it will be easy because the two sexes form different clusters.
#Except at smaller crabs where it will not be easy


Male <- rep(FALSE,n)

for(i in 1:n){
  if(as.numeric(crabdata$sex[i]) == 2){
    Male[i] <- TRUE
  }
}
#coef RW / CL = -2.71
#slope = 2.713, intercept = -2.940

logisticreg <- glm(sex ~ CL + RW, family = binomial(), data = crabdata)

logregclassifier <- rep(FALSE,n)
for(l in 1:n){
  if(logisticreg$fitted.values[l] > 0.5){
    logregclassifier[l] <- TRUE
  }
}

plot(crabdata$RW,crabdata$CL,pch="+",
     col=ifelse(!logregclassifier, "red","blue"),
     main=c("Logistic regression classified sex based on features",
            "Carapace Length and Rear Width"),ylab ="Carapace Length",
     xlab="Rear Width")

y <- c()
for( x in seq(from=5,to=22, by= 0.1)){
  y <- c(y,2.713 * x - 2.940)
}
lines(seq(from=5,to=22, by= 0.1),y)
legend("topleft",legend = levels(crabdata$sex),
       pch = "+",col=c("red","blue"))

LDA <- function(df){
```

```r
Male <- rep(FALSE,n)

for(i in 1:n){
  if(as.numeric(df$sex[i]) == 2){
    Male[i] <- TRUE
  }
}
Female <- !Male
Maleclass <- df[Male,5:6]
nMale <- nrow(Maleclass)
Maleproportion <- nMale / n
Femaleclass <- df[Female,5:6]
nFemale <- n - nMale
Femaleproportion <- 1 - Maleproportion

Malecenter <- 1/ nMale * colSums(Maleclass)
Femalecenter <- 1/ nFemale * colSums(Femaleclass)

male_sum_matr <- matrix(c(0,0,0,0),2)
female_sum_matr <- matrix(c(0,0,0,0),2)

# Male Female are actually 50/50, so nFemale = nMale

for(j in 1:nMale){
  male_sum_matr <- male_sum_matr +
    crossprod(as.matrix(Maleclass[j,]) - Malecenter,
              as.matrix(Maleclass[j,]) - Malecenter)
  female_sum_matr <- female_sum_matr +
    crossprod(as.matrix(Femaleclass[j,]) - Femalecenter,
              as.matrix(Femaleclass[j,]) - Femalecenter)
}
Malecov <- 1/ nMale * male_sum_matr
Femalecov <- 1/ nFemale * female_sum_matr

totalcov <- 1/2 * (Malecov + Femalecov)

totalobs <- df[,5:6]

maleclassifier <- rep(FALSE,n)
discriminantvalue <- rep(-100,n)
Maleconst <- -(1/2) * Malecenter %*% solve(totalcov) %*% Malecenter + log(Maleproportion)
Femaleconst <- -(1/2) * Femalecenter %*% solve(totalcov) %*% Femalecenter + log(Femaleproportion)

for(k in 1:n){
  malediscriminant <- as.numeric(totalobs[k,]) %*% solve(totalcov) %*% Malecenter
  femalediscriminant <- as.numeric(totalobs[k,]) %*% solve(totalcov) %*% Femalecenter
  discriminantvalue[k] <- malediscriminant + Maleconst - femalediscriminant - Femaleconst
  if(discriminantvalue[k] > 0){
    maleclassifier[k] <- TRUE
  }
}
```

```r
  print(solve(totalcov) %*% (Malecenter - Femalecenter))
  print(paste("Male constant:",Maleconst))
  print(paste("Female constant:",Femaleconst))
  return(maleclassifier)
}

#coef RW / CL = -2.9
maleclassifier <- LDA(crabdata)

plot(crabdata$RW,crabdata$CL,pch="+",
     col=ifelse(!maleclassifier, "red","blue"),
     main=c("LDA classified sex based on features",
            "Carapace Length and Rear Width"),ylab ="Carapace Length",
     xlab="Rear Width")
#intercept -5.066 slope 2.918

y <- c()
for( x in seq(from=5,to=22, by= 0.1)){
  y <- c(y,2.918 * x - 5.066)
}

lines(seq(from=5,to=22, by= 0.1),y)
legend("topleft",legend = levels(crabdata$sex),
       pch = "+",col=c("red","blue"))

library(XLConnect)
library(tree)
library(e1071)
wb = loadWorkbook("data/creditscoring.xls")
data = readWorksheet(wb, sheet = "credit", header = TRUE)
data$good_bad <- as.factor(data$good_bad)
head(data)
n <- dim(data)[1]

rows <- 1:n
test_index <-list()
set.seed(12345)
for(i in 1:4){
  test_index[[i]] <- sample(rows,250)

  rows <- setdiff(rows,test_index[[i]])
}

train=data[c(test_index[[1]],test_index[[2]]),]
valid=data[test_index[[3]],]
test = data[test_index[[4]],]
ntrain = dim(train)[1]
nvalid = dim(valid)[1]
ntest = dim(test)[1]

fittree1 <- tree(good_bad ~., data = train, split = "deviance")
fittree2 <- tree(good_bad ~., data = train, split = "gini")
```

```r
plot(fittree1)
text(fittree1,pretty=5)

plot(fittree2, type="uniform")
text(fittree2,pretty=5)

trainfit1= predict(fittree1, newdata=train, type="class")
testfit1 = predict(fittree1, newdata=test, type="class")

trainfit2= predict(fittree2, newdata=train, type="class")
testfit2 = predict(fittree2, newdata=test, type="class")

count_true <- function(a,b){
  counttrue <- 0
  for( i in 1:length(a)){
    if(a[i] == b[i]){
      counttrue <- counttrue + 1
    }
  }
  return(counttrue / length(a))
}

trainproptrue1 <- count_true(trainfit1,train$good_bad)
testproptrue1 <- count_true(testfit1,test$good_bad)

trainproptrue2 <- count_true(trainfit2,train$good_bad)
testproptrue2 <- count_true(testfit2,test$good_bad)

#fittree1 performs better than fittree2 on both training and test data.
#We choose to work with the split = "deviance"

paste("Misclassification rate of fittree1 for training data: ",1-trainproptrue1)
paste("Misclassification rate of fittree1 for test data: ",1-testproptrue1)
paste("Misclassification rate of fittree2 for training data: ",1-trainproptrue2)
paste("Misclassification rate of fittree2 for test data: ",1-testproptrue2)


# table(train$good_bad,trainfit1)
# table(test$good_bad,testfit1)
# table(train$good_bad,trainfit2)
# table(test$good_bad,testfit2)

#pruning fittree1

trainScore=rep(0,16)
testScore=rep(0,16)
for(i in 2:16) {
  prunedTree=prune.tree(fittree1,best=i)
  pred=predict(prunedTree, newdata=valid, type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}
plot(2:16, trainScore[2:16], type="b", col="red",
     main="Deviance of best pruned tree against no. of terminal nodes",
     xlab = "no. of terminal nodes",ylab ="Deviance",ylim=c(250,600))
points(2:16, testScore[2:16], type = "b" , col="blue")
legend("topright",c("training data","validation data"),
       lty=c(1,1),
       lwd=c(2.5,2.5),col=c("red","blue"))

#best tree appears to be size 8
# besttree <- prune.tree(fittree1,best=8)
# plot(besttree)
# text(besttree,pretty=5)
#this tree could just as well be size 5...deviance measure makes it misleading
besttree <- prune.tree(fittree1,best=5)
plot(besttree)
text(besttree,pretty=5)

#It seems having long duration loans, low levels of savings and being young
#makes one a bad loan taker, according to this tree (which it must be said is
#quite simplistic a view, since many "bad" are misclassified as "good")

predtestbest <- predict(besttree, newdata=test, type="class")
testbestproptrue <- count_true(predtestbest,test$good_bad)
paste("Misclassification rate of besttree for test data: ",1-testbestproptrue)
#table(test$good_bad,predtestbest)

nbayesmodel <- naiveBayes(good_bad~.,data=train)

nbayespredtrain<- predict(nbayesmodel, train)
nbayespredtest<- predict(nbayesmodel, test)

table(train$good_bad,nbayespredtrain)
table(test$good_bad,nbayespredtest)

nbayestrainproptrue <- count_true(nbayespredtrain,train$good_bad)
nbayestestproptrue <- count_true(nbayespredtest,test$good_bad)

paste("Misclassification rate of nbayesmodel for training data: ",1-nbayestrainproptrue)
paste("Misclassification rate of nbayesmodel for test data: ",1-nbayestestproptrue)
#Naive bayes classifies worse than the best decision tree, and unpruned tree.

raws <- data.frame(predict(nbayesmodel, newdata=train, type="raw"))

preds <- c()
for (i in 1:ntrain){
  if((raws[i,1]/raws[i,2]) > 0.1){
    preds[i] = "bad"
  }else{
    preds[i] ="good"
  }
}

table(train$good_bad, preds)
lossmatrnbayestrainproptrue <- count_true(preds,train$good_bad)
```

```r
paste("Misclassification rate of nbayesmodel with loss matrix for training data: ",
      1-lossmatrnbayestrainproptrue)

rawTest <- predict(nbayesmodel, newdata=test, type="raw")
predtests <- c()
for (i in 1:ntest){
  if((rawTest[i,1]/rawTest[i,2]) > 0.1){
    predtests[i] = "bad"
  }else{
    predtests[i] ="good"
  }
}

table(test$good_bad, predtests)
lossmatrnbayestestproptrue <- count_true(predtests,train$good_bad)
paste("Misclassification rate of nbayesmodel with loss matrix for test data: ",
      1-lossmatrnbayestestproptrue)
```