Introduction to Machine Learning

Lab 3

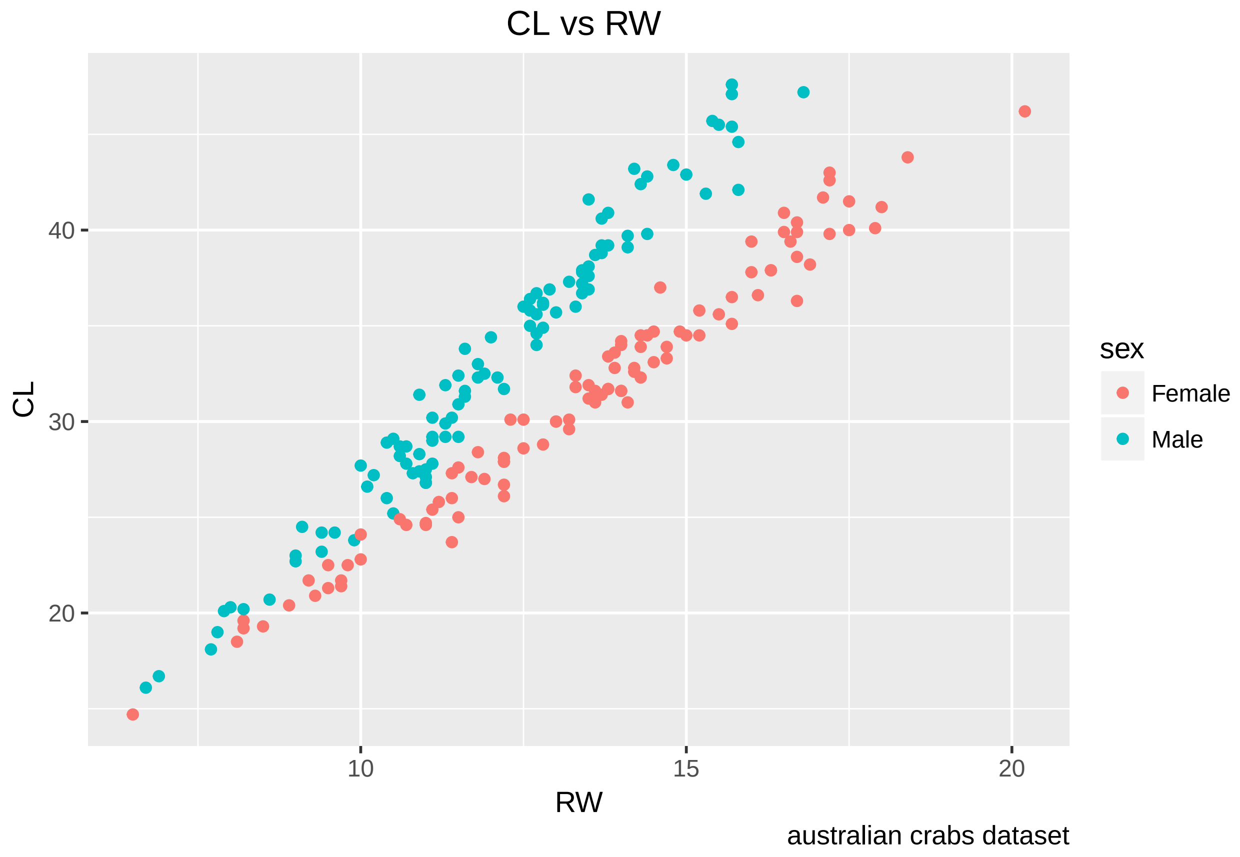
Anton Persson, Emil Klasson Svensson, Mattias Karlsson, Rasmus Holm

2016-11-26

Contents

Assignment 1

1.1



A line would be able to separate the genders pretty well. One could expect a couple of misclassifications when the variables has low values as the genders are not linearly separable there.

1.2

The return object of the LDA-function returns a list with all answers and the decision boundary.

To get the decision boundary we have the set the two discriminant functions equal to each other and solve for one of the parameters depending on X so that we get where they intersect.

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

$$w_k = \Sigma^{-1}\mu_k$$
$$w_{0k} = (-1/2)\Sigma^{-1}\mu_k + log(\pi_k)$$

$$\delta_{male}(x) = \delta_{female}(x)$$

$$\delta_{male}(x) - \delta_{female}(x) = 0$$

$$x^T(w_{Male} - w_{Female}) + (w_{0Male} - w_{0Female}) = 0$$

$$x_{CL}(w_{2Male} - w_{2Female}) + x_{RW}(w_{1Male} - w_{1Female}) + (w_{0Male} - w_{0Female}) = 0$$

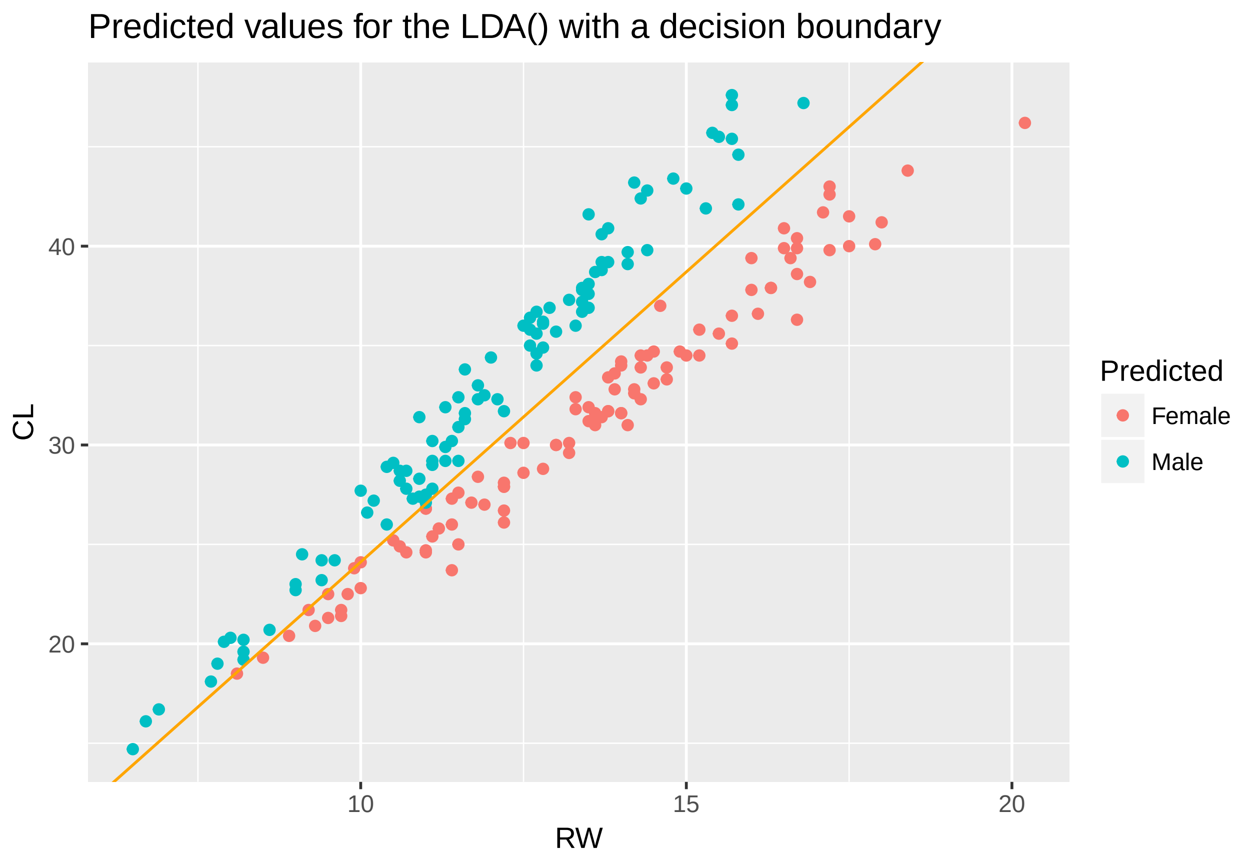
$$x_{RW}(w_{1Male} - w_{1Female}) + (w_{0Male} - w_{0Female}) = -x_{CL}(w_{2Male} - w_{2Female})$$

And we arrive the final result as.

$$x_{CL} = \frac{(x_{RW}(w_{1Male} - w_{1Female}) + (w_{0Male} - w_{0Female}))}{-(w_{2Male} - w_{2Female})}$$

This is expressed as y = CL and x = RW, which means this decision boundary needs to be plotted with the right variable on the right axis.

1.3



The line was calculated to

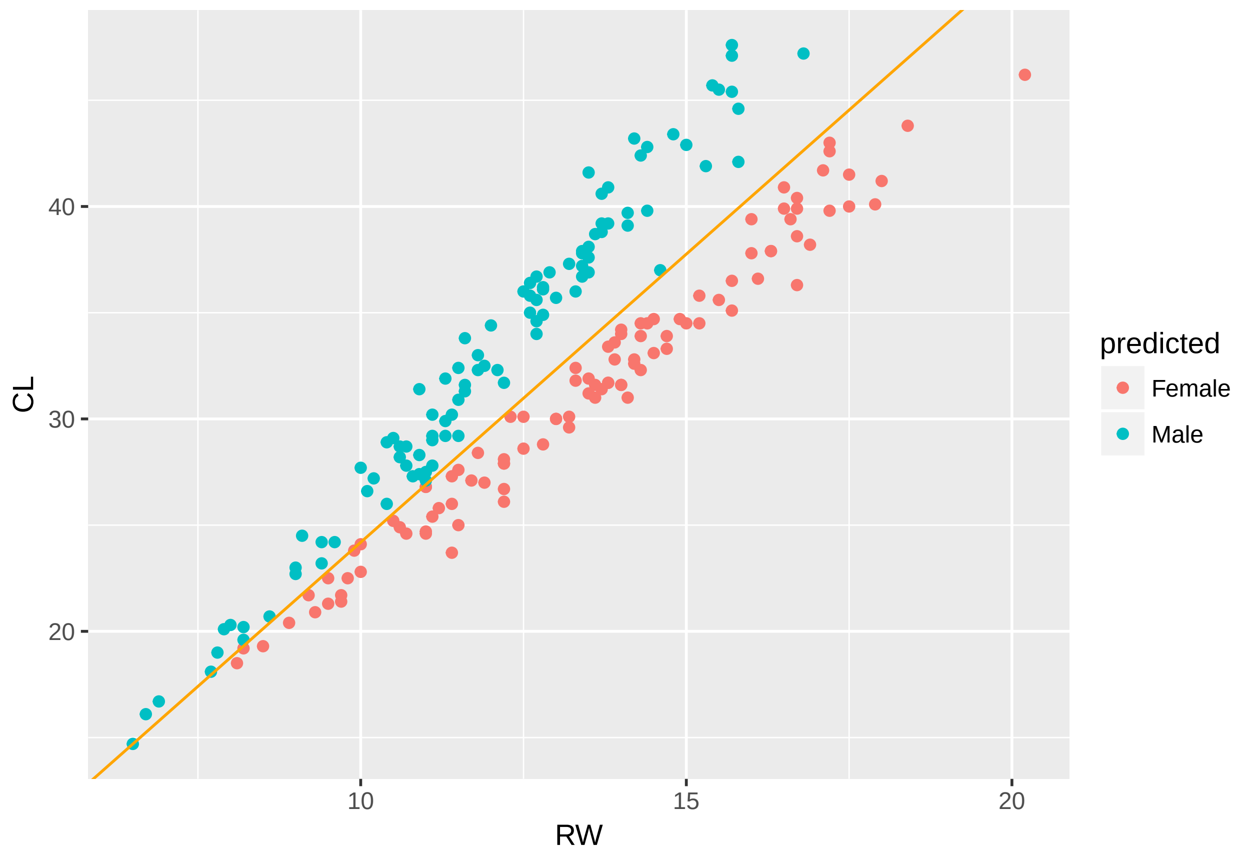
$$x_{CL} = -5.0656387 + 2.9180154 x_{RW}.$$

The decision line divides the data nicely but it has some issues when RW is below 12 where the two groups are closer in distance. This is as expected as we have already notice that the data is not linearly separable.

The two discriminant functions were calculated to

$$\delta_{male}(x) = (-12.5634175) + 2.5658514 x_{RW} + (-0.2138144) x_{CL},$$
$$\delta_{female}(x) = (-22.4287694) + 8.2486981 x_{RW} + (-2.1613185) x_{CL}.$$

1.4



One visible difference in the plots are that an observation located close to the line at CL = 37 and RW = 14 is now classified as a male whereas it was classified as a female. Other than this it is hard to distinguish any visible differences.

The line was calculated as follows

$$\frac{b_0 + b_{RW}}{b_{CL}}.$$

```
#> For the Logistic regression
#>         Predicted
#> Observed Female Male
#>   Female     97    3
#>   Male        4   96
#> For the LDA:
#>         Predicted
#> Observed Female Male
#>   Female     97    3
#>   Male        4   96
```

Both classifiers has the same misclassification rate 7/200 and has the same amount (but not necessery the same) of misclassifications in the anti-diagonals for the different categories.

Assignment 2

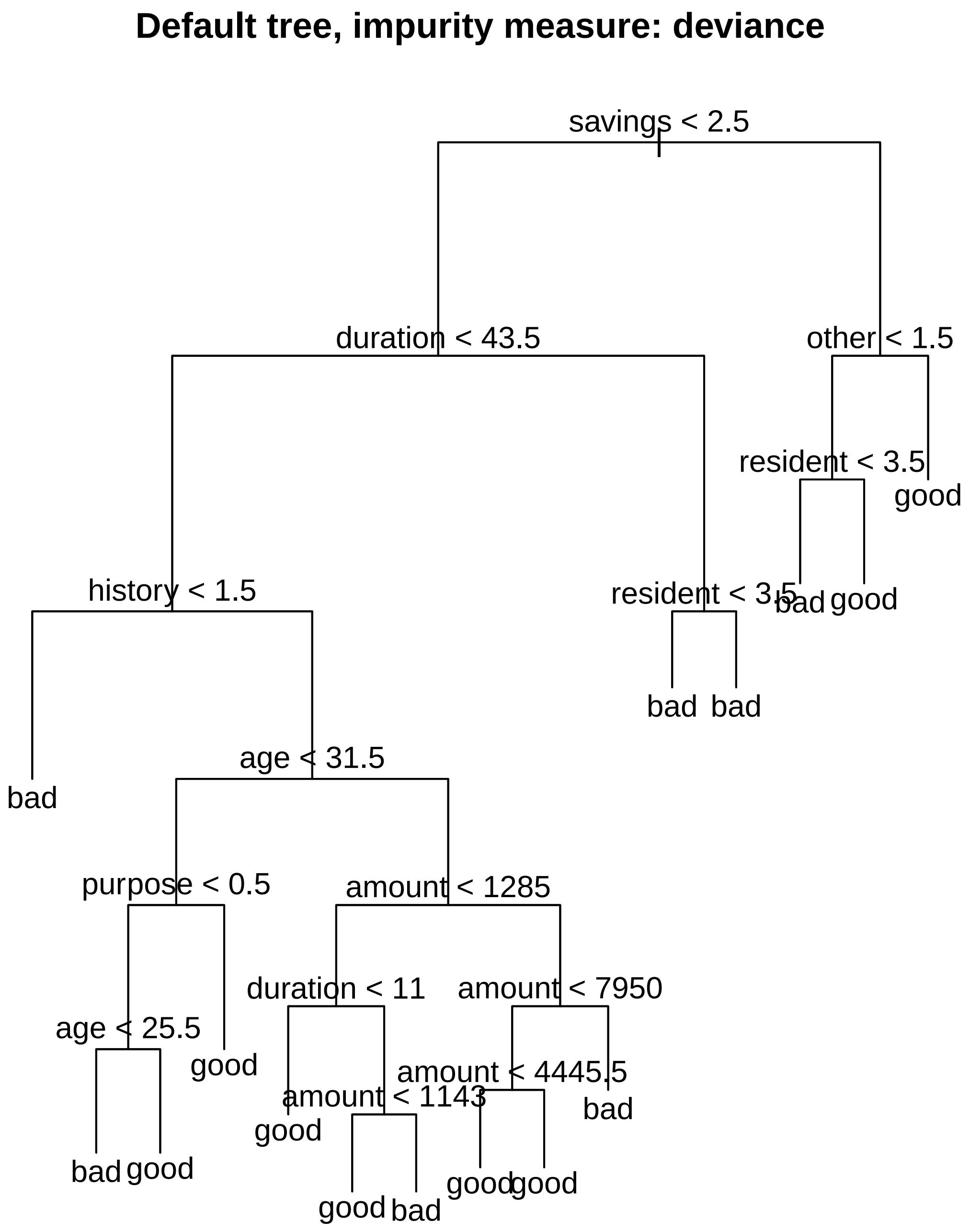
2.1

We used the following R code to separate the data into training-, validation- and test set.

```r
#2.1
credit<-read.xls("../data/creditscoring.xls")
set.seed(12345)
samples<-1:nrow(credit)
train_id<-sample(samples, floor(nrow(credit)*.5))
samples2<-samples[-train_id]
valid_id<-sample(samples2, floor(nrow(credit)*.25))
test_id<-samples[-c(train_id, valid_id)]
train<-credit[train_id, ]
validation<-credit[valid_id,]
test<-credit[test_id,]
```

2.2a Deviance

The default decision tree, created by the tree function, with *deviance* as impurity measure is shown below.



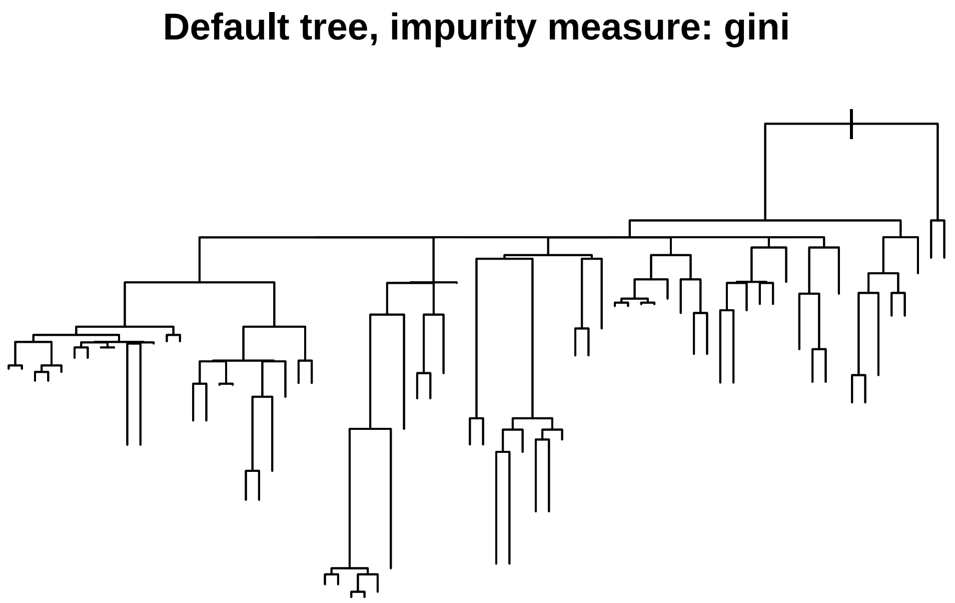
The actual tree and their rules are shown above. The misclassification rates for both the training- and test data are shown below.

```
#> $Training_error
#> [1] 0.212
#>
#> $Test_error
#> [1] 0.284
```

The error rate in the training data is about 21%, about 28% for the test data.

2.2b Gini

The same procedure for the decision tree with *gini* as impurity measure is presented below.



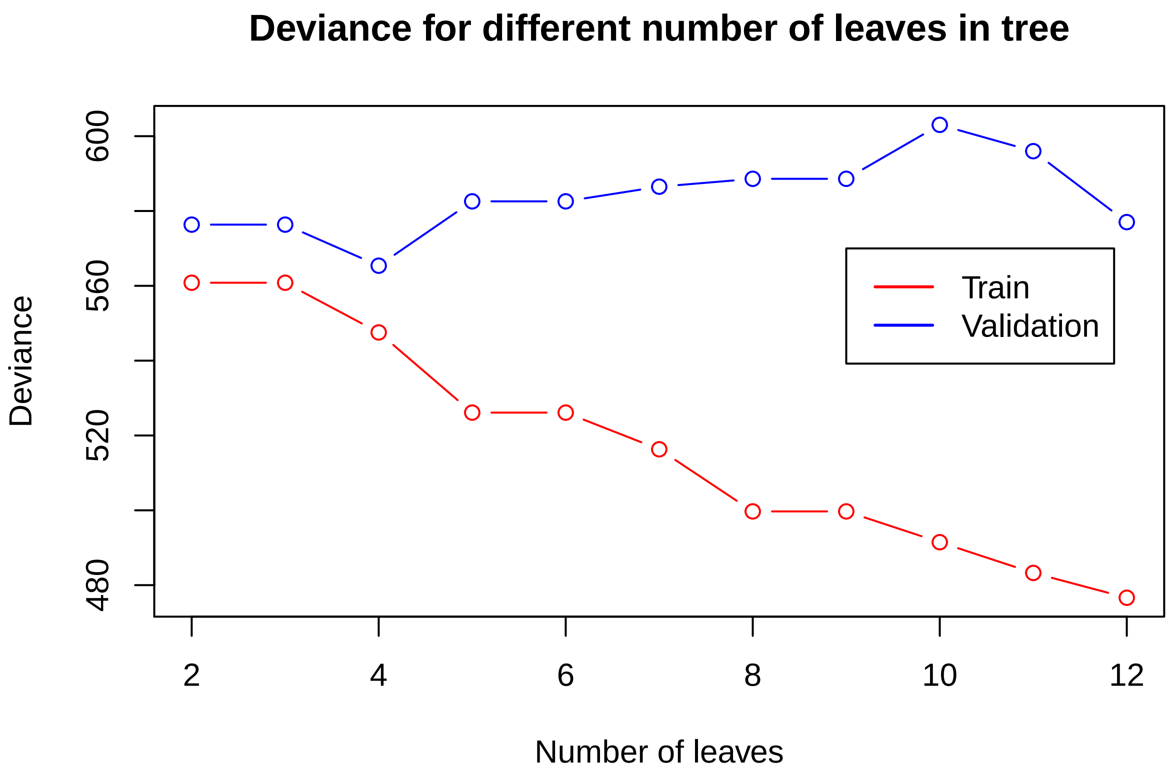
Since this tree is more complex, it has 72 leaves (compared to 15 in the deviance-tree) it is not possible to view the rules as previously. The missclassification rates for both data sets are as follows.

```
#> $Training_error
#> [1] 0.23
#>
#> $Test_error
#> [1] 0.34
```

The error rate for the gini tree is 23 % on the training set, and 34 % on the test set. That's worse than the respective rates for the deviance tree. Therefore, we will be using the deviance as the impurity measure from now on.

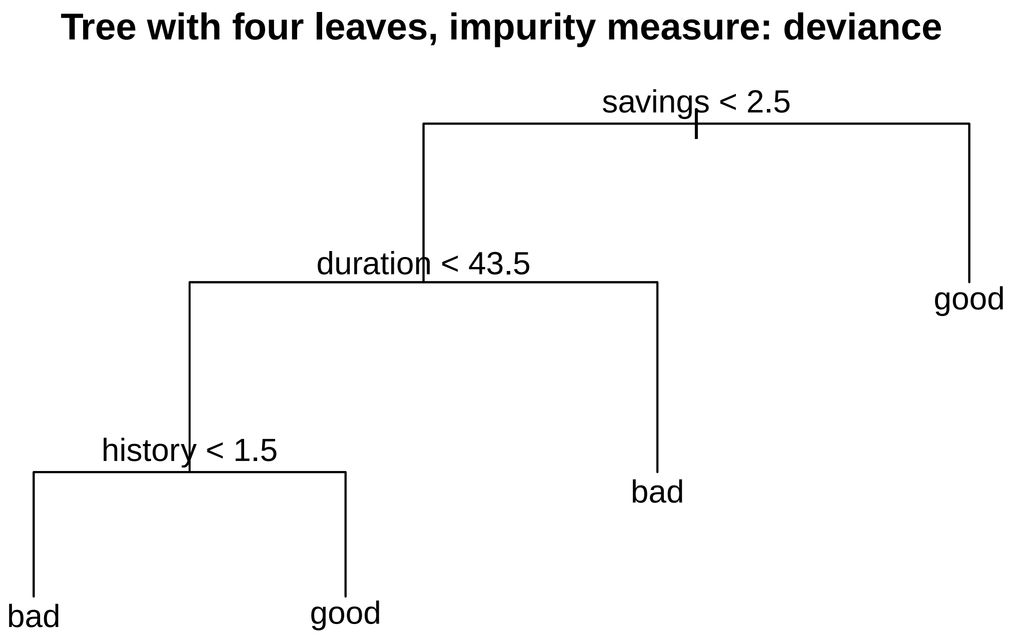
2.3

The required plot of the dependencies of deviances for different number of leaves, for both the training and validation data, is presented below.



It should be noted that we multiplied the deviance for the validation set predictions by 2, since the train data set is twice as big. This was done since the deviance() function only summarizes the deviance. That action isn't necessary, the same result will come out of the analysis but it may be easier to analyze. However, from looking at the plot above, we can conclude that four leaves seems to result in the optimal tree since the validation error is at its lowest peak there. The deviance of the training monotonically decreases which indicates that the model starts to overfit as the number of leaves increases.

The optimal tree is visualized below.



The tree above has the depth three. It has four leaves, as concluded earlier, and the three variables used in the construction of the tree are *Savings, Duration* and *History.* In general, the tree says that if a potential customer has big savings, he or she will pay back the loan. We don't know exactly what the other variables mean so we cannot make any conclusions as to whether they make sense or not. The misclassification for the tree on the test data is presented below.

```
#> $Missclassification_rate_testdata
#> [1] 0.26
```

The misclassification rate has sunk from 28.6% in the first tree model (with 15 leaves), to 26% in this model. So the model has become better at classifying the test data and is also much easier to interpret as one would have hoped.

2.4

The requested confusion matrices and their respective error rates are presented below.

```
#> $confusion_matrix_train
#>       truth
#> pred   bad good
#>   bad   95   98
#>   good  52  255
#>
#> $error_rate_train
#> [1] 0.3
#>
#> $confusion_matrix_test
#>       truth
#> pred   bad good
#>   bad   47   49
#>   good  31  123
#>
#> $error_rate_train
#> [1] 0.32
```

From the missclassification errors we can conclude that this naive bayes model has worse error rates than the tree in assignment 2.3.

2.5

In this assignment we made some changes in the loss function. Practically, what we have done is to recognize that it's expensive to classify a customer as "good" when the truth is that the customer is "bad", i.e. false positive, at managing their loans. This means that to make the classification "good", the probability of the customer getting classified as "good" must be 10 times bigger than the probability for getting classified as "bad". The confusion matrices for the training- and test data are shown below.

```
#> $confusion_matrix_train
#>       truth
#> pred   bad good
#>   bad  137  263
#>   good  10   90
#>
#> $confusion_matrix_test
#>       truth
#> pred   bad good
#>   bad   70  131
#>   good   8   41
```

From the confusion matrices above we conclude that the error rates get worse, which is reasonable. Since the classifier really don't want to classify a customer as "good" when he or she really turns out to be "bad". It classifies too many observations as "bad", a lot more false negatives than before, which results in an increase of misclassifications in total.

In this assignment, the vast majority of my misclassifications are classified as "bad" but truth is "good". In assignment 2.4, the proportions between the two different error rates are much more equal. And this is actually a good thing in this case, since the task is about to really avoid to classify customers as "good" in managing their loans, when they turn out to be "bad", i.e. not pay back the money.

Appendix

Code for Assignment 1

```r
crabs <- read.csv("../data/australian-crabs.csv")
library(ggplot2)
p<- ggplot(data = crabs) + geom_point(aes(x = RW, y =CL, col = sex)) +
    labs(title = " CL vs RW", caption = "australian crabs dataset")
plot(p)


LDA<- function(X){

    RW <- X[,1]
    CL <- X[,2]
    sex<- X[,3]
    myMu <- aggregate(cbind(RW,CL),by = list(sex), FUN = mean, simplify = TRUE)
    myCov <- by(cbind(RW,CL), list(sex), cov, method = "pearson")
    myPi <- aggregate(cbind(RW,CL),by = list(sex),
                      FUN =function(x) length(x)/nrow(cbind(RW,CL)), simplify = TRUE)

    mySig<- (( myCov[[1]] * myPi[2,2] * length(RW) ) +
             (myCov[[2]] * myPi[2,3] * length(RW)) ) / nrow(X)

    woMale <- -0.5 * as.matrix(myMu[2,2:3],ncol = 2) %*% solve(mySig) %*%
        t(myMu[2,2:3]) + log(myPi[2,3])
    woFem <- -0.5 * (as.matrix(myMu[1,2:3],ncol = 2)) %*% solve(mySig) %*%
        t(myMu[1,2:3]) + log(myPi[1,3])

    wM<- solve(mySig) %*% t(myMu[2,2:3])
    wF<- solve(mySig) %*% t(myMu[1,2:3])


    a <- (woMale - woFem)
    b <- wM - wF
    x <- cbind(X[,1:2])

    myInter <- as.numeric(-a/b[2])
    mySlope <- as.numeric(-b[1]/b[2])

    X$myClass<-t(ifelse((a[1] + t(b) %*% t(x)) > 0 ,levels(X[,3])[2],levels(X[,3])[1]))
    colnames(X)[4] <- "Predicted"
    retObj<-list(w0 = c(woMale,woFem),
                 w1 = cbind(wM=wM,wF=wF),
                 myClass = X,
                 myModel = c(myInter = myInter, mySlope = mySlope))

    return(retObj)
}

results <- LDA(crabs[,c(5,6,2)])

ggplot(data = results$myClass) +
    geom_point(aes(x = RW, y =CL, col = Predicted)) +
    geom_abline(intercept = results$myModel[1],
                slope = results$myModel[2], col = "orange") +
    labs(title = "Predicted values for the LDA() with a decision boundary")

myLogit<-glm(sex~RW + CL, family = binomial(link='logit'), data = crabs )

myDecLog<-coef(myLogit)[1:2]/-coef(myLogit)[3]

predicted <- factor(ifelse(myLogit$fitted.values > 0.5,1,0),
                    levels=c(0, 1), labels=c("Memale", "Male"))

ggplot(data = results$myClass) +
    geom_point(aes(x = RW, y =CL, color = predicted)) +
    geom_abline(intercept = myDecLog[1], slope = myDecLog[2], color="orange")

cat("For the Logistic regression \n")

t(table(Predicted = ifelse(myLogit$fitted.values > 0.5,"Male","Female"),
      Observed = crabs$sex))
cat("\n")
cat("For the LDA:\n")
t(table(Predicted = results$myClass[,4] , Observed = crabs$sex))
```

Code for Assignment 2

```
#2.1
credit<-read.csv2("../data/creditscoring.csv", header=T, sep=";")
set.seed(12345)
samples<-1:nrow(credit)
train_id<-sample(samples, floor(nrow(credit)*.5))
samples2<-samples[-train_id]
valid_id<-sample(samples2, floor(nrow(credit)*.25))
#any (id2 %in% id) FALSE
test_id<-samples[-c(train_id, valid_id)]
train<-credit[train_id, ]
validation<-credit[valid_id,]
test<-credit[test_id,]

#2.2a
library(tree)
dev_tree<-tree(formula = good_bad~., split="deviance", data=train)

plot(dev_tree)
text(dev_tree, pretty=0)
title("Default tree, impurity measure: deviance")

#Train
dev_fit_train<-predict(dev_tree, newdata=train, type="class")

table_dev_train<-table(pred=dev_fit_train, truth=train$good_bad)
```

```
error_dev_train<-1-sum(diag(table_dev_train))/sum(table_dev_train)

#Test
dev_fit_test<-predict(dev_tree, newdata=test, type="class")

table_dev_test<-table(pred=dev_fit_test, truth=test$good_bad)

error_dev_test<-1-sum(diag(table_dev_test))/sum(table_dev_test)

dev_errors<-list(Training_error=error_dev_train, Test_error=error_dev_test)
dev_errors

#2.2b
gini_tree<-tree(formula = good_bad~., split="gini", data=train)

plot(gini_tree)
title("Default tree, impurity measure: gini")

#2.2 gini train
gini_fit_train<-predict(gini_tree, newdata=train, type="class")

table_gini_train<-table(pred=gini_fit_train, truth=train$good_bad)

error_gini_train<-1-sum(diag(table_gini_train))/sum(table_gini_train)

#Test
gini_fit_test<-predict(gini_tree, newdata=test, type="class")

table_gini_test<-table(pred=gini_fit_test, truth=test$good_bad)
error_gini_test<-1-sum(diag(table_gini_test))/sum(table_gini_test)

gini_errors<-list(Training_error=error_gini_train, Test_error=error_gini_test)
gini_errors

#2.3 med deviance då.

train_score<-rep(0,12)
valid_score<-rep(0,12)
for(i in 2:length(train_score)) {
  pruned_tree<-prune.tree(dev_tree,best=i)
  pred<-predict(pruned_tree, newdata=validation,
                type="tree")
  train_score[i]<-deviance(pruned_tree)
  valid_score[i]<-deviance(pred)*2
  #eftersom deviance returnerar total deviance, valid hälften så stor
}
plot(x=2:length(train_score), y=train_score[-1], type="b",
     col="red", ylim=c(min(train_score[-1]),
      max(valid_score)), xlab="Number of leaves", ylab="Deviance")
points(x=2:length(valid_score), y=valid_score[-1], type="b", col="blue")
title("Deviance for different number of leaves in tree")
legend(9,570, c("Train","Validation"),col=c("red","blue"),  lty=c(1,1), lwd=c(1.5,1.5))
```

```r
dev_tree<-tree(formula = good_bad~., split="deviance", data=train)

tree_4leaves<-prune.tree(dev_tree, best=4)

plot(tree_4leaves)
text(tree_4leaves, pretty=0)
title("Tree with four leaves, impurity measure: deviance")

dev_fit_final<-predict(tree_4leaves, newdata=test,
             type="class")

table_dev_final<-table(pred=dev_fit_final, truth=test$good_bad)

error_dev_final<-1-sum(diag(table_dev_final))/sum(table_dev_final)
final_tree_error<-list(Missclassification_rate_testdata=error_dev_final)
final_tree_error


#2.4
library(e1071)

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

#Train
nb_fit_train<-predict(nb, newdata=train, type="class")
#sum(nb_fit_train=="good")
nb_table_train<-table(pred=nb_fit_train, truth=train$good_bad)

error_nb_train<-1-(sum(diag(nb_table_train))/sum(nb_table_train))


#Test
nb_fit_test<-predict(nb, newdata=test, type="class")
nb_table_test<-table(pred=nb_fit_test, truth=test$good_bad)

error_nb_test<-1-(sum(diag(nb_table_test))/sum(nb_table_test))

naive_bayes<-list(confusion_matrix_train=nb_table_train, error_rate_train=error_nb_train,
      confusion_matrix_test=nb_table_test, error_rate_train=error_nb_test)

naive_bayes

#2.5 train
nb_fit_raw_train<-predict(nb, newdata=train, type="raw")

nb_fit_raw_train<-as.data.frame(nb_fit_raw_train)
nb_fit_raw_train$class[nb_fit_raw_train$good>(nb_fit_raw_train$bad*10)]<-"good"
nb_fit_raw_train$class[nb_fit_raw_train$good<=(nb_fit_raw_train$bad*10)]<-"bad"
nb_fit_raw_train$class<-as.factor(nb_fit_raw_train$class)

nb_loss_train<-table(pred=nb_fit_raw_train$class, truth=train$good_bad)

#2.5 test
```

```
nb_fit_raw_test<-as.data.frame(predict(nb, newdata=test, type="raw"))

nb_fit_raw_test$class[nb_fit_raw_test$good>(nb_fit_raw_test$bad*10)]<-"good"
nb_fit_raw_test$class[nb_fit_raw_test$good<=(nb_fit_raw_test$bad*10)]<-"bad"
nb_fit_raw_test$class<-as.factor(nb_fit_raw_test$class)

nb_loss_test<-table(pred=nb_fit_raw_test$class, truth=test$good_bad)

nb_loss_confusion<-list(confusion_matrix_train=nb_loss_train, confusion_matrix_test=nb_loss_test)
nb_loss_confusion
```

Contributions

We divided the work into two parts and discussed/compiled the results in pairs. Then we all discussed our findings together as a whole group and checked that everyone had similar/understood the results.