TDDE01 – Machine Learning
Individual Lab Report 3

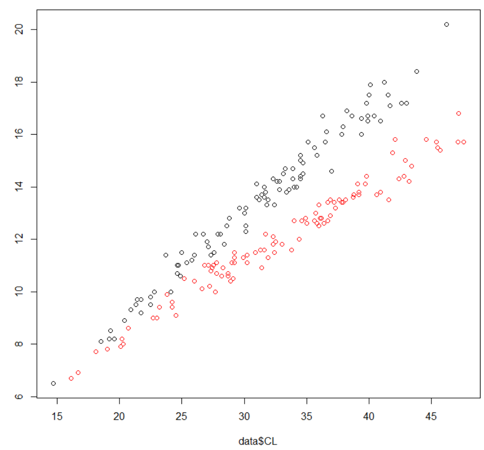
Martin Estgren `<mares480>`

26 november 2016

1 Assignment 1

This assignment involves creating an implementation of the *linear discriminant analysis* (LDA) with *maximum likelihood estimation* (MLE) as *discrimination function*. The LDA function will then be used to classify the *sex* of Chicago crabs using the *carapace length* and the *rear width* from the data set given for the assignment.

The first step involves visual inspection of the data set to determine if a *linear discrimination function* would be a good fit for the data set. The result of the *response variable sex* is ploted as the color of each point with the predictors *carapace length* and *rear width* as X and Y dimensions.



Figur 1: Raw plot of the data set.

Visual inspection of the data set indicates that a linear classification model would be suitable.

In order to implement the LDA function we start by implementing the MLA and return the weights for each classification category.

$$w_{1i} = -\frac{1}{2}\mu_i^T \Sigma^{-1} \mu_i + log\pi_i \tag{1}$$

where $i \in \{CL, RW\}$

$$b_{1i} = \Sigma^{-1} \mu_i \tag{2}$$

The classification function can be derived form these variables by combining them into:

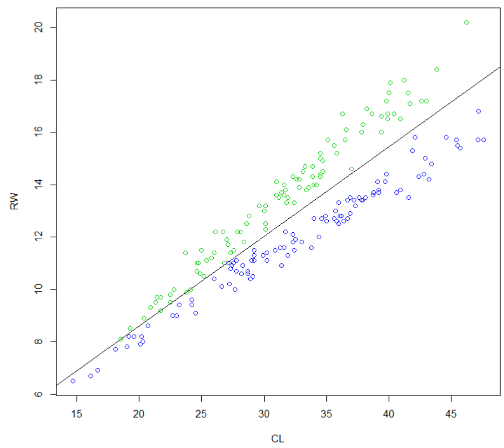
$$d_i(x) = X^T \Sigma^{-1} \mu_i - \frac{1}{2}\mu_i^T \Sigma^{-1} \mu_i + log\pi_i \tag{3}$$

In our case we have two categories for the classifier: $\{Male, Female\}$ which means that we need to combine the two different $d_i(x)$ functions with $d(x) = d_{Male}(x) - d_{Female}(x)$. We are in addition to classifying the data points interested in drawing the LDA line in the plot. In order to do this the intercept and slope of the LDA function is calculated from the $d(x)$ in the following way:

$$intercept = \frac{-b_1}{w_{1RW}} \tag{4}$$

$$slope = \frac{-w_{1CL}}{w_{1RW}} \tag{5}$$

The resulting plot can be seen below

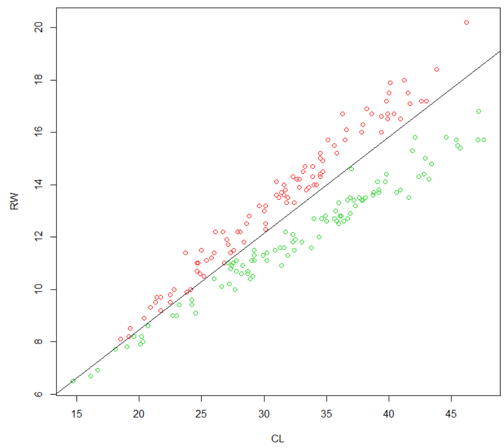


Figur 2: Plot of the raw data classified with LDA.

The misclassification rate is 0 meaning the result is a close to perfect classification.

As observed in above, the LDA managed to perform a perfect classification. The next task involves using the built in *logistic regression* classifier and examine its performance on the same data set.

Once again we extract the coefficients from the discriminant model and plot both the predicted result together with the *discrimination bound*.



Figur 3: Plot of the raw data classified with logistic regression.

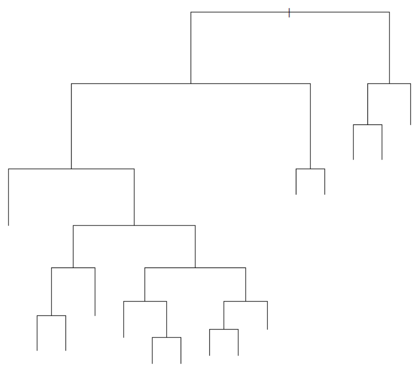
The *logistic regression* also managed a similar classification of the given data set with.

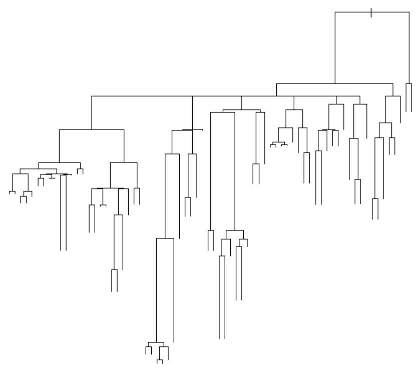
2 Assignment 2

In this assignment we are tasked with finding a classification model to find potential good customers who can manage their loans in a goodåay. To achieve this we use a data set with observations about how previous customers has manged their loans given a set of predictors.

We start by splitting the observations into 50,25,25 percent parts and try fitting a decision tree model based on *deviance* and *gini index*. Below are the confusion matrices and misclassification rates for the testing and training observations.

Below the two different types of description trees can be observed followed by the confusion matrices and miss classification rate.



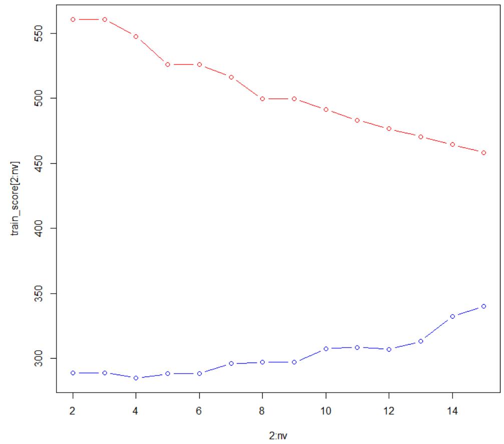


The *deviance* metric created a much less complex tree compared to the *gini index*. This is mirrored in the misclassification ate where the *deviance* creates the best result.

```
deviance model fitness
       predicted
       bad good
  bad   29   17
  good  45  159
misclassification rate: 0.248
```

```
gini model fitness
       predicted
       bad good
  bad   24   26
  good  50  150
misclassification rate: 0.304
```

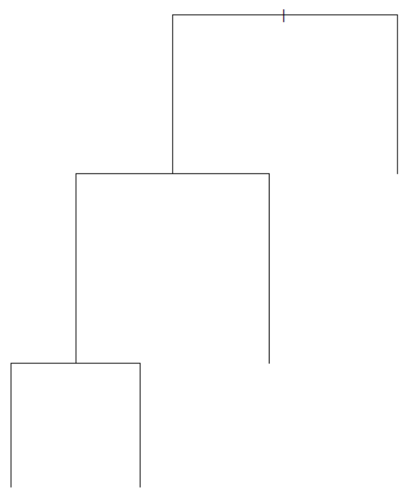
In order to determine the best max tree depth of the *deviance* model we iterates through all depths between 2 and 15.



Figur 4: deviance for different max tree depths.

The *red* line indicates deviance of the training data and the *blue* line, the validation data. Tree depth of 4 results in the least deviance for the validation data.

```
optimal depth deviance fitness
       predicted
       bad good
  bad   20   59
  good   9  162
misclassification rate: 0.272
```



Figur 5: Optimal tree model.

The following confusion matrix represents the same data set used on a *naive bayesian* model instead of a decision tree.

```
model fitness of the training set
      predicted
      bad good
bad   95   98
good  52  255
misclassification rate: 0.3

model fitness of the testing set
      predicted
      bad good
bad   52   53
good  22  123
misclassification rate: 0.3
```

We can observe about the same misclassification ratio as the decision tree model. The training data predictions have about the same level of misclassification as the testing set but with a higher level of true positives.

We now apply a *loss matrix* to the predictions of the model and compare the result to the result above. The loss matrix looks the following:

$$\begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix} \tag{6}$$

```
model fitness of the training set
      predicted
truth bad good
bad   27   17
good  120  336
misclassification rate: 0.274

model fitness of the testing set
      predicted
truth bad good
bad  18  10
good 56  166
misclassification rate:0.264
```

The result indicates a improvement in performance of about 2-3 percent.

3 Appendix: A - Code assignment 1

Listing 1: Code for assignment 1

```
library(readxl)

data = read.csv("australian-crabs.csv")
set.seed(12345)

plot(data$CL,data$RW, col=data$sex)
# Is the data easy to calssify be linear
   discriminant analysis?
# : Yes, very easy
X = cbind(data$CL,data$RW)
Y = data$sex
#ASSIGNMENT 1.2

disc_fun=function(label, S){
  X1=X[Y==label,]
```

```
  #MISSING: compute LDA parameters w1 (vector with 2
      values) and w0 (denoted here as b1)
  estimated_prob =  nrow(X1) / nrow(X)
  estimated_mean = colMeans(X1)
  b1 = -0.5*t(estimated_mean)%*%solve(S)%*%estimated
     _mean+log(estimated_prob)
  w1 = solve(S)%*%estimated_mean
    return(c(w1[1],w1[2],b1[1,1]))
}

X1=X[Y=="Male",]
X2=X[Y=="Female",]

S=cov(X1)*dim(X1)[1]+cov(X2)*dim(X2)[1]
S=S/dim(X)[1]

#discriminant function coefficients
res1=disc_fun("Male",S)
res2=disc_fun("Female",S)
print(res1)
print(res2)
# MISSING: use these to derive  decision boundary
   coefficients 'res'
res = res1-res2
intercept = -res[3] / res[2]
slope = -res[1]/res[2]
print(intercept)
print(slope)
# classification
d=res[1]*X[,1]+res[2]*X[,2]+res[3]
Yfit=(d>0)
plot(X[,1], X[,2], col=Yfit+3, xlab="CL", ylab="RW")
#MISSING: use 'res' to plot decision boundary.
abline(intercept,slope)

model = glm( sex ~ CL + RW, data = data, family = "
   binomial")
res_log = coefficients(model)
d_log = res_log[2]*X[,1]+res_log[3]*X[,2]+res_log[1]
Yfit_log=(d_log>0)
plot(X[,1], X[,2], col=Yfit_log+2, xlab="CL", ylab="
```

```
   RW")
print(res_log)
intercept_log = -res_log[1]/res_log[3]
slope_log = -res_log[2]/res_log[3]
abline(intercept_log,slope_log)
```

4 Appendix: B - Code assignment 2

Listing 2: Code for assignment2

```
library(readxl)
library(tree)
library(e1071)
library(rpart)

data = read.csv("creditscoring.csv")
#data$good_bad = as.characterdata$good_bad == "good"
n = nrow(data)
set.seed(12345)

indexes = sample(1:n,n)
end_traning = floor(n*0.5)
end_validation = end_traning + floor(n*0.25)

traning_indexes = indexes[1:end_traning]
validation_indexes = indexes[(end_traning+1):end_
   validation]
testing_indxes = indexes[(end_validation+1):n]

train = data[traning_indexes,]
validation = data[validation_indexes,]
testing = data[testing_indxes,]

dtreefit <- tree(as.factor(good_bad) ~ ., data=train
   , split = c("deviance"))
gtreefit <- tree(as.factor(good_bad) ~ ., data=train
   , split = c("gini"))

d_yfit = predict(dtreefit, newdata = testing,type="
   class")
g_yfit = predict(gtreefit, newdata = testing,type="
```

```
   class")
plot(dtreefit)
plot(gtreefit)

d_table = table(d_yfit,testing$good_bad)
g_table = table(g_yfit,testing$good_bad)

print(d_table)
print(1-sum(diag(d_table))/sum(d_table))
print(g_table)
print(1-sum(diag(g_table))/sum(g_table))

nv = summary(dtreefit)[4]$size
train_score = rep(0,nv)
test_score = rep(0,nv)
for(i in 2:nv){
  pruned=prune.tree(dtreefit,best=i)
  pred=predict(pruned, newdata=validation, type="
     tree")
  train_score[i] = deviance(pruned)
  test_score[i] = deviance(pred)
}
plot(2:nv,train_score[2:nv], col="Red",type = "b",
   ylim=c(min(test_score[2:nv]),max(train_score)))
points(2:nv,test_score[2:nv],col="Blue",type="b")

final = prune.tree(dtreefit,best=4)
yfit = predict(final,newdata=validation,type="class"
   )
f_table = table(validation$good_bad,yfit)
print(f_table)
print(1-sum(diag(f_table))/sum(f_table))
plot(final)

bayes_model = naiveBayes(good_bad ~., data=train)

test_yfit = predict(bayes_model, testing[,-ncol(
   testing)], type = "class")
train_yfit = predict(bayes_model, train[,-ncol(train
   )])
```

```
naive_table = table(test_yfit,testing$good_bad)
naive_table_train = table(train_yfit,train$good_bad)


print(naive_table_train)
print(1-sum(diag(naive_table_train))/sum(naive_table
   _train))

print(naive_table)
print(1-sum(diag(naive_table))/sum(naive_table))

# With loss matrix
bayes_model = naiveBayes( good_bad ~ ., data = train
   )

test_yfit = predict(bayes_model, testing[,-ncol(
   testing)],type="raw")
train_yfit = predict(bayes_model, train[,-ncol(train
   )], type="raw")

test_yfit =  (test_yfit[, 2] / test_yfit[, 1]) > 1/
   10
train_yfit =  (train_yfit[, 2] / train_yfit[, 1]) >
   1/10

naive_table = table(test_yfit,testing$good_bad)
naive_table_train = table(train_yfit,train$good_bad)

print(naive_table_train)
print(1-sum(diag(naive_table_train))/sum(naive_table
   _train))

print(naive_table)
print(1-sum(diag(naive_table))/sum(naive_table))
```