Introduction to machine learning 732A95

Lab 2 block 1

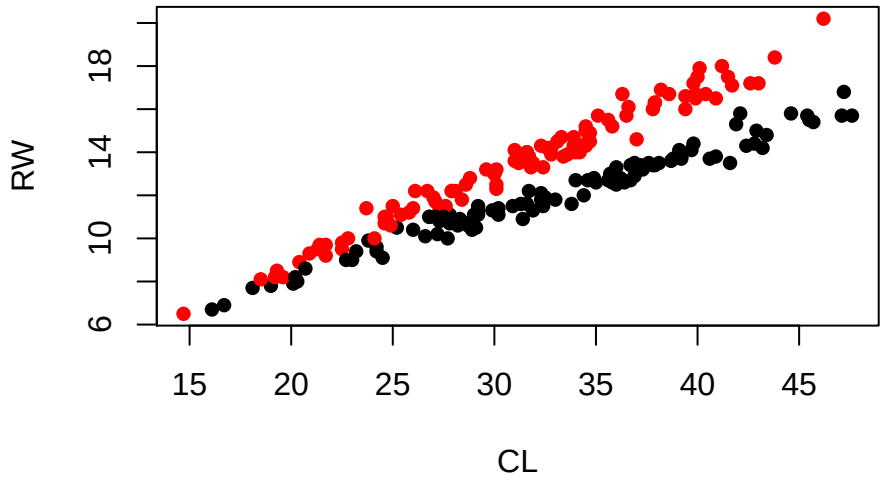
Eric Herwin
erihe068

Contents

Assignment 1. LDA and logistic regression

1.1

In this task I will use `australian-crabs.csv` and make a scatterplot of carapace length (CL) versus rear width (RW).



In the plot we can see the male (black) and female (red) crabs. We can see that when CL increases, there is a much more clearer that there is groups in a data. The LDA will have a much easier time fitting the groups there, but the groups are not that clear when CL is low.

1.2

In this task I will implement a LDA with proportional priors, inputs RW and CL and output Sex. I will then use this implementation to classify the observations, extract extract the discriminant functions and equation of the decision boundary.

The discriminant functions:

```
#Discriminant function for male
res1
```

```
##          RW          CL
##   2.5658514  -0.2138144 -12.5634175
```

```
#Discriminant function for female
res2
```

```
##          RW          CL
##   8.248698  -2.161318 -22.428769
```

For male:

$y_{male} = -12.56 + 2.565RW - 0.214CL$

For female:

$y_{female} = -22.42 + 8.25RW - 2.16CL$

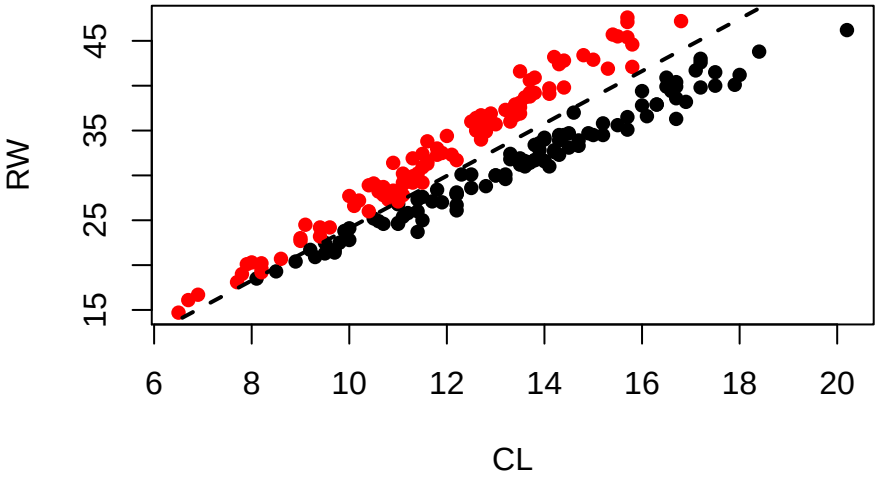
The decision boundary:

$$\beta_{0m} + \beta_{1m}RW + \beta_{2m}CL = \beta_{0f} + \beta_{1f}RW + \beta_{2f}CL$$

$$RW = \frac{\beta_{0f} + \beta_{2f}CL - (\beta_{0m} + \beta_{2m}CL)}{\beta_{1m} - \beta_{1f}} = \frac{-9.87 - 2.38CL}{-5.68} = 1.73 + 0.42CL$$

1.3

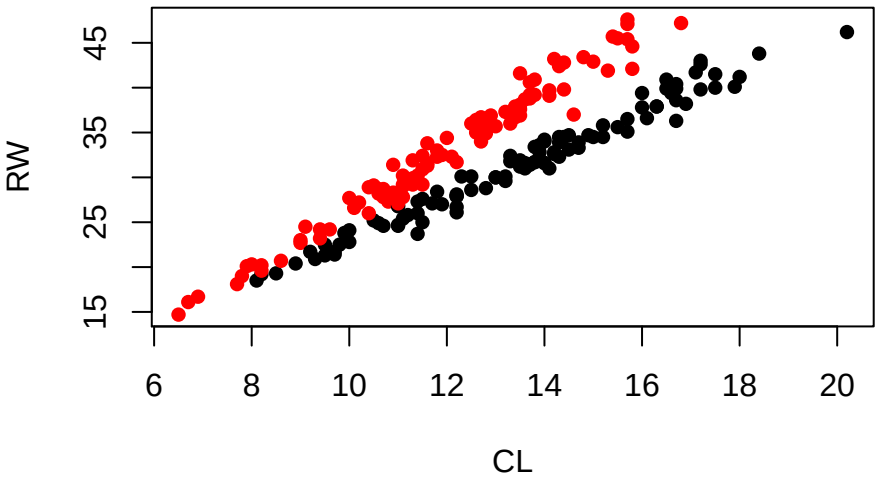
For this task I will make a plot of the original data with the decision boundary.



We can see that the model classify good, but worth a note is that in the data there is only obvious groups when CL increases. The LDA managed to classify the data good, but mostly due to this clear obvious groups when CL increases.

1.4

In this task I will do a simmilar classification by using a logistic regression.



We can see that the classification is similar to the LDA. We can conlude that for this data there is no clear difference between these two models (in terms of classification).

The equation function for the glm is:

$$y_i = \frac{e^{l_i}}{e^{l_i} + 1}$$

where

$$l_i = 13.617 - 12.564RW + 4.631CL + \epsilon$$

Assignment 2. Analysis of credit scoring

2.1

I will split up the data in to train, validation and test.

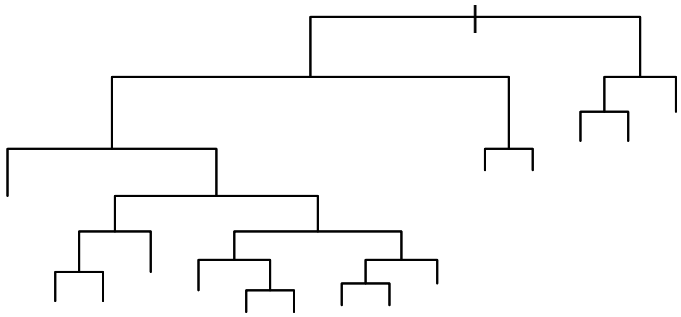
```
n <- dim(creditscoring)[1]
set.seed(12345)
id <- sample(1:n, n)
train <- creditscoring[id[1:500],]
validation <- creditscoring[id[501:750],]
test <- creditscoring[id[751:1000],]
```

2.2

In this task I will fit a desicion tree to the training data set and report the trees and missclassification rate.

a)

Deviance:

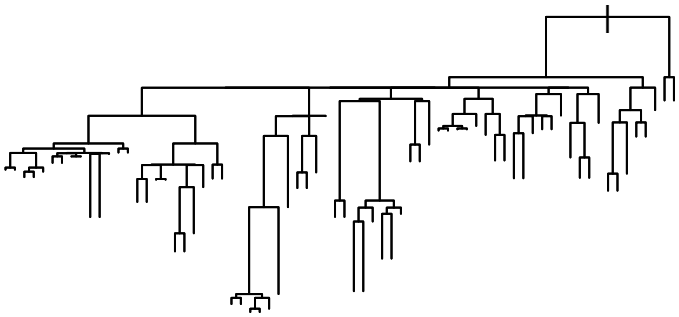


The missclassification rate:

```
## [1] 0.248
```

b)

Gini:



The missclassification rate:

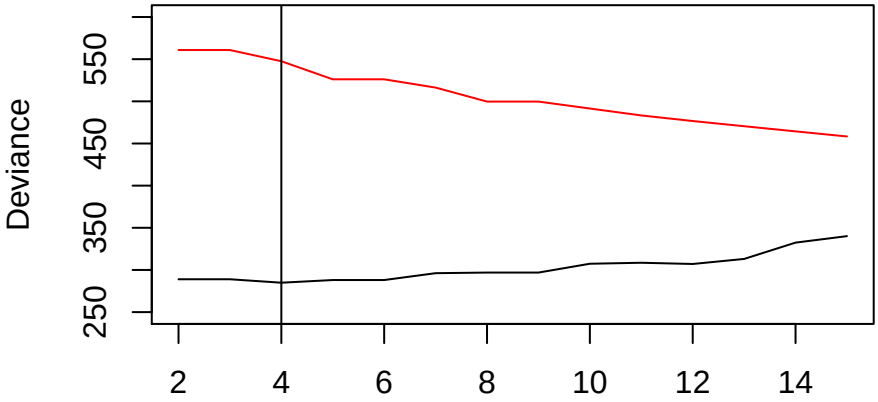
```
## [1] 0.304
```

Comparing the missclassifications between the two measures of impurity we can see that the `deviance` are the best. By looking at the structure of the trees, the gini is creating alot more leaves, therefore making it more overfitted and complex then the tree with `deviance`.

2.3

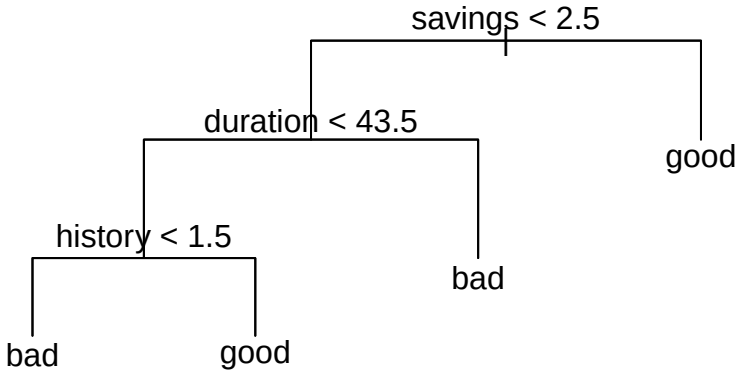
In this task I will use the training and validation sets to choose the optimal tree depth.

The plot shows how the deviance is with different amount of leaves.



We can see that the most optimal depth of the tree is 4, where the red line is the training deviance and the black is the prediction deviance.

The most optimal tree is computed:



We can see from the output which variables are choosen are: savings, duration and history.

```
## [1] 0.248
```

We can see that the missclassification rate for the test data is 0.3.

2.4

In this task I will use the training data to perform a classification with Naive Bayes.

This is the output of the missclassification rate for the test data:

```
##       fit_test
##        bad good
##   bad   52   22
##   good  53  123
```

```
## [1] 0.3
```

This is the output of the missclassification rate for the test data:

```
##       fit_train
##        bad good
##   bad   95   52
##   good  98  255
```

```
## [1] 0.3
```

We can see that the missclassification rate for both the training and test is the same. Comparing to the previous step we can see that the missclassification rate, for the test data, are lower for the classification tree. We can conclude that the tree is better for this data.

2.5

In this task I will again compute a Naive Bayes for the data, but using a different loss-matrix for the `good_bad` varaible.

Training:

```
conf2.5_tr
```

```
##       new_fit
##        bad good
##   bad   27  120
##   good  17  336
```

```
missclass_naive_tr
```

```
## [1] 0.274
```

Test:

```
conf2.5
```

```
##       new_fit
##        bad good
##   bad   18   56
##   good  10  166
```

```
missclass_naive
```

```
## [1] 0.264
```

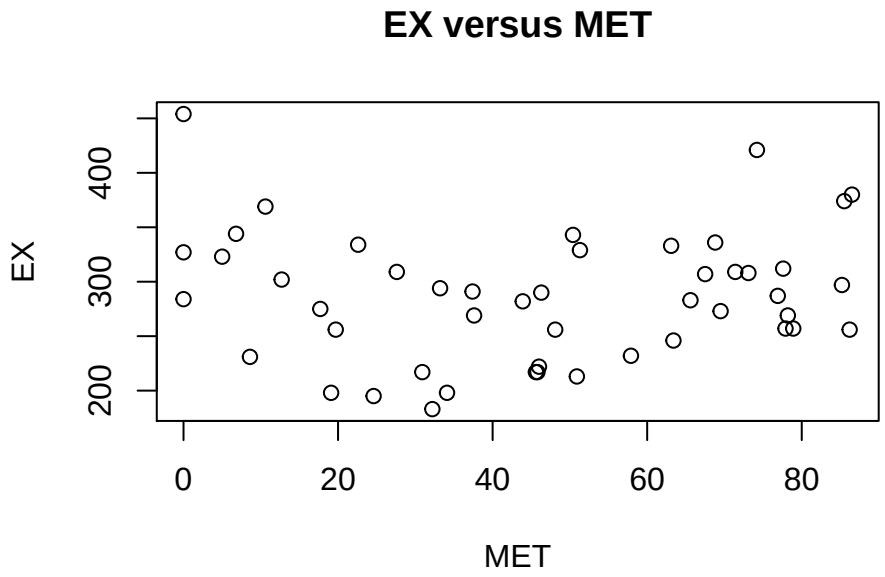
The loss-matrix given in the task tells us that it costs us more to underestimate the data. In other words, put more weight for the overestimation. In the confusion matrices we can see that the model actually overestimates much more then it underestimates. The logical thing is that when we force a model to under or overestimate, the missclassification rate will be generally higher, but for this estimation the missclassification rate becomes much lower then the previous step. This might be because the method tries to fit the most optimal solution and it might missclassify a part of some group. And when I affect this optimal soloution, the result might be that the groups are classified more correctly.

Assignment 3. Uncertainty estimation

3.1

In this task I will just reorder the data provide a plot for EX versus MET.

```
State <- State[order(State$MET),]
plot(y = State$EX, x = State$MET, xlab = "MET", ylab = "EX", main = "EX versus MET")
```



An appropriate model would be something that could take the curved shape of the data in to consideration.

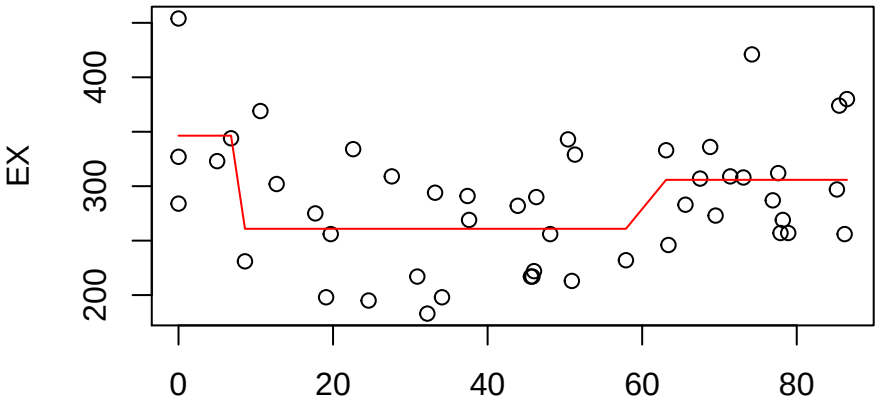
3.2

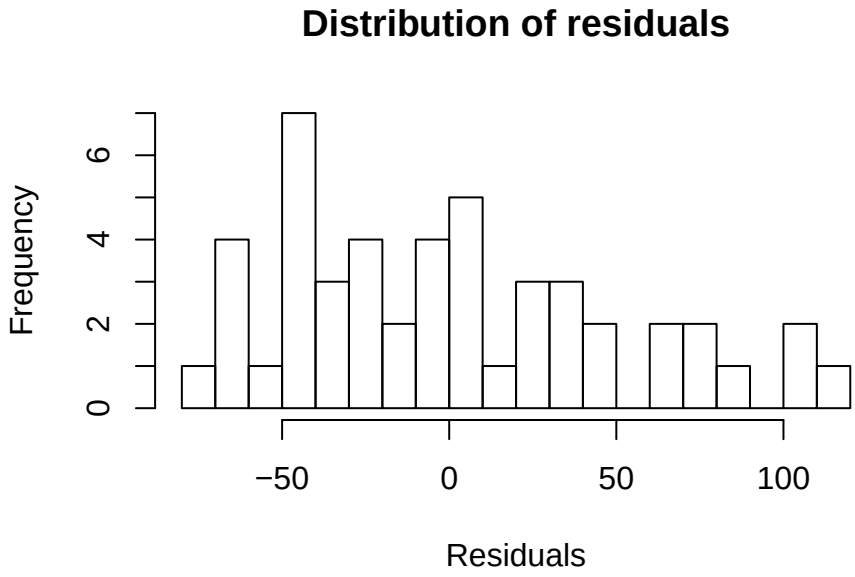
In this tak I will use the package `tree` and fit a regression tree model with target EX and feature MET.

```
## $size
## [1] 5 3 1
##
## $dev
## [1] 176390.1 164923.6 191417.2
##
## $k
## [1]      -Inf  4504.356 20538.320
##
## $method
## [1] "deviance"
```

```
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

From the `cv.tree()` output I can conclude that the one with the lowest deviance is with a depth of 3.



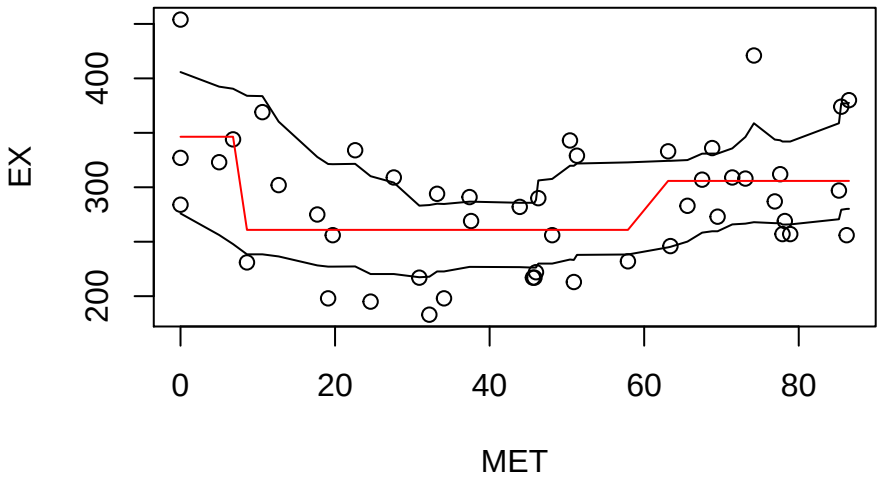


We can see that the fit that the model is giving (red) is fitting the data that fits the data in a accepteble way, but not that smooth. This is because the regression tree is fitting the data into 3 parts.

For the distribution of the residuals we can see that there is a large variance where the distribution does not resemble a normal distribution. We can also see from the fitting of the model on to data that there will be a large residual variance.

3.3

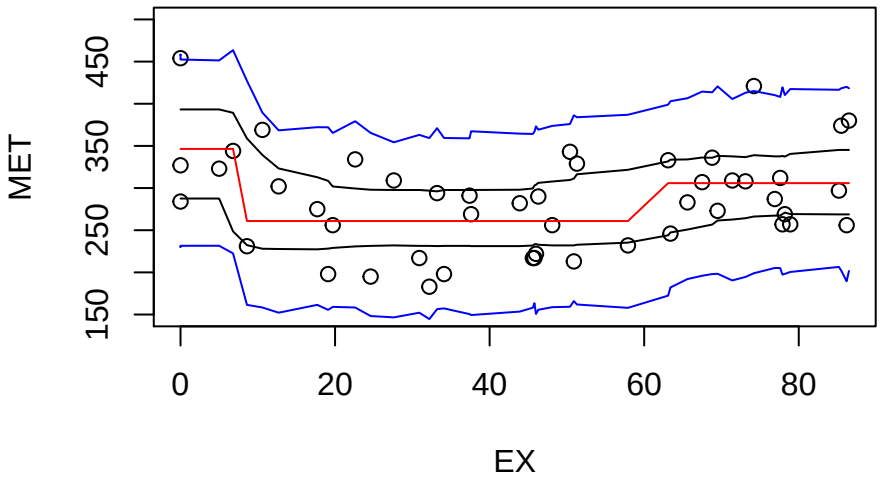
In this step I will compute and plot the 95% confidence bands for the regression tree model from step 2 by using a non-parametric bootstrap.



We can see that the CI bands (black) are a bit bumpy, but are following the data better then the fit. The CI are covering a good amount of the observations, but are not that smooth. The CI seems to be very dependent on the induvidual observations.

3.4

In this task I will compute and plot the 95% confidence and prediction bands the regression tree model from step 2 by using a parametric bootstrap.



We can see that the CI band (black) is much more smoothened compared to the non-parametric bootstrap. This CI is not that dependant on the induvidual observations and therefore the more smoothened lines.

We can also observe that outside the 95% PI bands there is not 5% of the data outside, but since the data set are very small we can conclude that there is not enough observations to observe this at this time. The PI tells us that 95% of an infinite amount of future observations will fall inside the bands, given the current model.

3.5

Going with a tree regression we got the histogram of the residuals from step 2. We concluded that the histogram had a large variance and it did not resemble a normal distubution. Since the nonparametric bootstraps make no assumptions about how the observations are distributed, the assumption of normal distributed residuals is not then required.

Appendix

```
#ASSINGMENT 1

## ---- echo=FALSE, message=FALSE-----------------------------------------
australian_crabs <- read_csv("/home/eric/Documents/732A95/ML_LAB2/australian-crabs.csv")
#australian_crabs <- read_csv("C:/Users/Eric/OneDrive/Dokument/732A95/ML_LAB2/australian-crabs.csv")

australian_crabs <- data.frame(australian_crabs)


## -----------------------------------------------------------------------
plot(australian_crabs$CL, australian_crabs$RW, ylab = "RW", xlab= "CL")

## ----echo=FALSE---------------------------------------------------------
Y <- australian_crabs$sex
X <- data.frame(RW = australian_crabs$RW, CL = australian_crabs$CL)
pi_k <- c(table(Y)[1]/length(Y), table(Y)[2]/length(Y))


bound <- "Male" == Y
male_x <- X[bound,]
bound <- "Female" == Y
female_x <- X[bound,]
mu_kMale <- apply(male_x, 2, mean)
mu_kFemale <- apply(female_x, 2, mean)

mu_list <- list(mu_kMale = mu_kMale, mu_kFemale= mu_kFemale)

S <- cov(male_x)*dim(male_x)[1] + cov(female_x)*dim(female_x)[1]
S <- S/dim(X)[1]

w_0i_male <- -(1/2)*t(mu_list[[1]]) %*% solve(S) %*% mu_list[[1]] + log(pi_k[1])
w_i_male <- solve(S) %*% mu_list[[1]]

w_0i_female <- -(1/2)*t(mu_list[[2]]) %*% solve(S) %*% mu_list[[2]] + log(pi_k[2])
w_i_female <- solve(S) %*% mu_list[[2]]

fit_male <- c(t(w_i_male) %*% t(X) + c(w_0i_male))
fit_female <- c(t(w_i_female) %*% t(X) + c(w_0i_female))


res1 <- c(w_i_male[1,1], w_i_male[2,1], w_0i_male[1])
res2 <- c(w_i_female[1,1], w_i_female[2,1], w_0i_female[1])

res <- res1-res2

d=res[1]*X[,1]+res[2]*X[,2]+res[3]
Yfit=(d>0)

sseq_test <- seq(6,20, length.out = 200)

#the line is where the lise is
```

```
line <- c()
for(rew in 1:length(sseq_test)){
  vec <- res[1]*sseq_test[rew]
  line[rew] <- polyroot(c(res[3]+ vec, res[2]))
}




## ---- echo=FALSE----------------------------------------------------------
plot(X[,1], X[,2], col=Yfit+1, xlab="CL", ylab="RW", pch = 16)
lines(y=line,x= sseq_test, lwd=2, lty = 2)

## ---- message=FALSE-------------------------------------------------------
glm_log <- glm(factor(Y) ~ ., data=X, family = binomial())
d <- predict(glm_log)
Yfit=(d>0)

plot(X[,1], X[,2], col=Yfit+1, xlab="CL", ylab="RW", pch = 16)



#ASSIGNMENT 2

## ---- echo=FALSE----------------------------------------------------------
creditscoring <- read_excel("/home/eric/Documents/732A95/ML_LAB2/creditscoring.xls")
#creditscoring <- read_excel("C:/Users/Eric/OneDrive/Dokument/732A95/ML_LAB2/creditscoring.xls")
creditscoring <- data.frame(creditscoring)

## -------------------------------------------------------------------------
n <- dim(creditscoring)[1]
set.seed(12345)
id <- sample(1:n, n)
train <- creditscoring[id[1:500],]
validation <- creditscoring[id[501:750],]
test <- creditscoring[id[751:1000],]


## ---- echo=FALSE----------------------------------------------------------
trad_deviance <- tree(as.factor(good_bad) ~., data=train, split = "deviance")
plot(trad_deviance)
text(trad_deviance, pretty = 0)

## ---- echo=FALSE----------------------------------------------------------
pred <- predict(trad_deviance, test, type = "class")
#0 = bad, 1= good
confusion_matr <- table(test$good_bad, pred)
missclass_dev <- (confusion_matr[1,2] + confusion_matr[2,1])/sum(confusion_matr)
missclass_dev

## ---- echo=FALSE----------------------------------------------------------
trad_gini <- tree(as.factor(good_bad) ~., data=train, split = "gini")
plot(trad_gini)
```

```
## ---- echo=FALSE-----------------------------------------------------------
pred <- predict(trad_gini, test, type = "class")
confusion_matr <- table(test$good_bad, pred)
missclass_gini<- (confusion_matr[1,2] + confusion_matr[2,1])/sum(confusion_matr)
missclass_gini

## ---- echo=FALSE-----------------------------------------------------------
trad_gini3 <- tree(as.factor(good_bad) ~., data=train, split = "deviance" )

trainScore=rep(0,15)
testScore=rep(0,15)

for(i in 2:15) {
  prunedTree=prune.tree(trad_gini3,best=i)
  pred=predict(prunedTree, newdata=validation, type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}



b_tree <- prune.tree(trad_gini3,best=4)
p_red <- predict(b_tree, newdata=test, type="class")
confusion_matr <- table(test$good_bad, p_red)
missclass_dev <- (confusion_matr[1,2] + confusion_matr[2,1])/sum(confusion_matr)
missclass_dev



fit <- naiveBayes(formula = as.factor(good_bad) ~., data=train)

fit_test <- predict(fit, test)
fit_train <- predict(fit, train)

conf2.4 <- table(test$good_bad, fit_test)
missclass_dev <- (conf2.4[1,2] + conf2.4[2,1])/sum(conf2.4)

conf2.4_tr <- table(train$good_bad, fit_train)
missclass_dev_tr <- (conf2.4_tr[1,2] + conf2.4_tr[2,1])/sum(conf2.4_tr)




L <- matrix(c(0,10,1,0),2)

#test
bayes_fit2 <- predict(fit, newdata = test, type="raw")

bayes_fit2[,1] <- L[2,1]*bayes_fit2[,1,drop=FALSE]


new_fit <- character(nrow(bayes_fit2))
```

```r
new_fit[bayes_fit2[,1]>bayes_fit2[,2]] <- "bad"
new_fit[new_fit == ""] <- "good"

conf2.5 <- table(test$good_bad, new_fit)
missclass_naive <- (conf2.5[1,2] + conf2.5[2,1])/sum(conf2.5)


#train
bayes_fit2 <- predict(fit, newdata = train, type="raw")

bayes_fit2[,1] <- L[2,1]*bayes_fit2[,1,drop=FALSE]

new_fit <- character(nrow(bayes_fit2))

new_fit[bayes_fit2[,1]>bayes_fit2[,2]] <- "bad"
new_fit[new_fit == ""] <- "good"

conf2.5_tr <- table(train$good_bad, new_fit)
missclass_naive_tr <- (conf2.5_tr[1,2] + conf2.5_tr[2,1])/sum(conf2.5_tr)




#ASSSINGMENT 3


## ---- echo=FALSE----------------------------------------------------------
#State <- read.csv2("~/Documents/732A95/ML_LAB2/State.csv")
State <- read.csv2("C:/Users/Eric/OneDrive/Dokument/732A95/ML_LAB2/State.csv")

## ----fig.width=5, fig.height=3.5,fig.align='center'-----------------------
State <- State[order(State$MET),]
plot(y = State$EX, x = State$MET, xlab = "MET", ylab = "EX", main = "EX versus MET")

## ---- echo=FALSE----------------------------------------------------------
tr3 <- tree(EX ~ MET, data = State )
set.seed(12345)
cv.tree(tr3, tree.control(nobs = nrow(State), minsize = 8))

## ---- echo=FALSE----------------------------------------------------------


optim_tree <- prune.tree(tr3,best=3)

plot(y = State$EX, x = State$MET, xlab = "EX", ylab = "MET")
points(y = predict(optim_tree), x = State$MET, col = "red", type = "l")

resi <-  State$EX - predict(optim_tree, newdata = State)

hist(resi, breaks = 10)
```

```
## ---- echo=FALSE-----------------------------------------------------------

f <- function(data, ind  ){
  data1=data[ind  ,]#    extractbootstrapsample
  res <- tree(EX ~ MET, data = data1 ,control =  tree.control(nobs = nrow(data1), minsize = 8)) #fit li
  optim_tree_f <- prune.tree(res, best=3)
  #predictvaluesfor all Area valuesfrom the original data
  priceP=predict(optim_tree_f,newdata=State)
  return(priceP)
}

bot <- boot(data = State, statistic = f, R=1000)
ci_boot <- envelope(bot)$point

plot(y = State$EX, x = State$MET, xlab = "EX", ylab = "MET")
points(x = State$MET, ci_boot[1,], col = "black", type = "l")
points(x = State$MET, ci_boot[2,], col = "black", type = "l")
points(x = State$MET, f(State, 1:48), col = "red", type = "l")


## ---- echo=FALSE-----------------------------------------------------------
mle <- prune.tree(tr3,best=3)

rng <- function(data, mle){
  data1=data.frame(EX=data$EX, MET=data$MET)
  n=length(data$EX)#generate new Price
  resid <- data1$EX - predict(mle, newdata=data1)
  data1$EX <- rnorm(n, predict(mle, newdata=data1), sd(resid))
  return(data1)
}


#CI
f1 <- function(data1){
  res=tree(EX ~ MET, data = data1,control =  tree.control(nobs = nrow(data1), minsize = 8)) #fit model
  optim_tree_f <- prune.tree(res, best=3)
  #predictvaluesfor all Area valuesfrom the original data
  priceP=predict(optim_tree_f,newdata=State)
  return(priceP)
}

res <- boot(State, statistic=f1, R=1000, mle=mle,ran.gen=rng, sim="parametric")
ci_boot_par <- envelope(res)$point



#PI
f1_pi <- function(data1){
  res=tree(EX ~ MET, data = data1) #fit model
  optim_tree_f <- prune.tree(res, best=3)
  #predictvaluesfor all Area valuesfrom the original data
  priceP <- predict(optim_tree_f,newdata=State)
  n <- length(State$EX)
```

```
  resid <- data1$EX - predict(mle, newdata=data1)
  predictedP=rnorm(n, priceP, sd(resid))
  return(predictedP)
}

p_boot <- boot(State, statistic=f1_pi, R=1000, mle=mle,ran.gen=rng, sim="parametric")

ci_boot_pi <- envelope(p_boot)$point

plot(y = State$EX, x = State$MET, xlab = "EX", ylab = "MET", ylim = c(150,500))
points(x = State$MET, ci_boot_par[1,], col = "red", type = "l")
points(x = State$MET, ci_boot_par[2,], col = "red", type = "l")
points(x = State$MET, ci_boot_pi[1,], col = "blue", type = "l")
points(x = State$MET, ci_boot_pi[2,], col = "blue", type = "l")
```