Lab 11

Attila Lazar

13.01.2021

a)

We read the data file and create a train and test dataset. we then train a tree on the train data.

```
data <- read.csv2("data/winequality-red.csv", na.strings="", dec=".", skipNul=TRUE)
str(data)
```

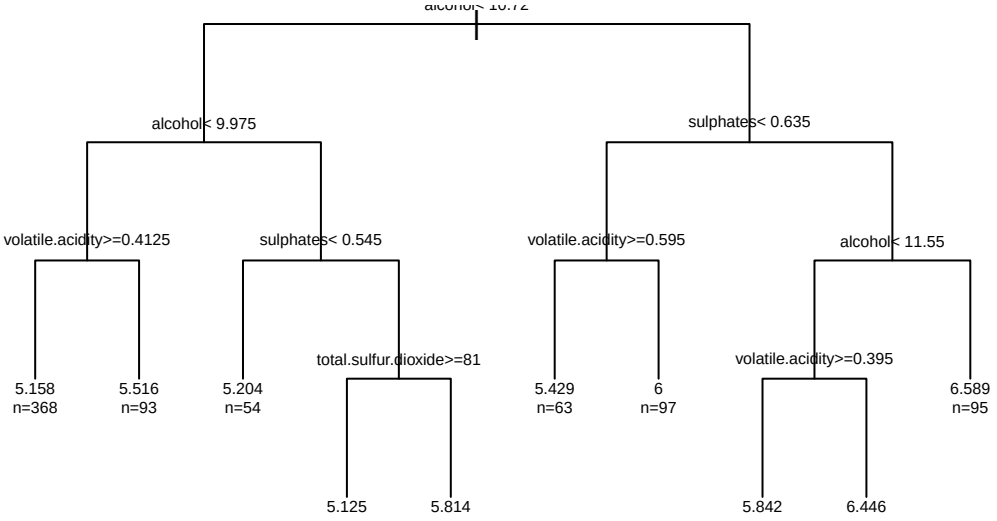
```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

```
set.seed(1234)
n <- nrow(data)
train <- sample(1:n, round(n*2/3))
test <- (1:n) [-train]

#install.packages("rpart")
library(rpart)

t0 <- rpart(quality~., data, subset=train)
```

b)

The structure of the trained tree. Alcohol seems to bee the most important variable.

```
plot(t0, uniform=TRUE, branch=1, compress=TRUE)
text(t0, use.n = TRUE, cex=0.5)
```



c)

We compute the RMSE on the test data

```
sqrt(mean((data[test, 'quality'] - predict(t0, data[test,]))^2))
```

```
## [1] 0.7009297
```

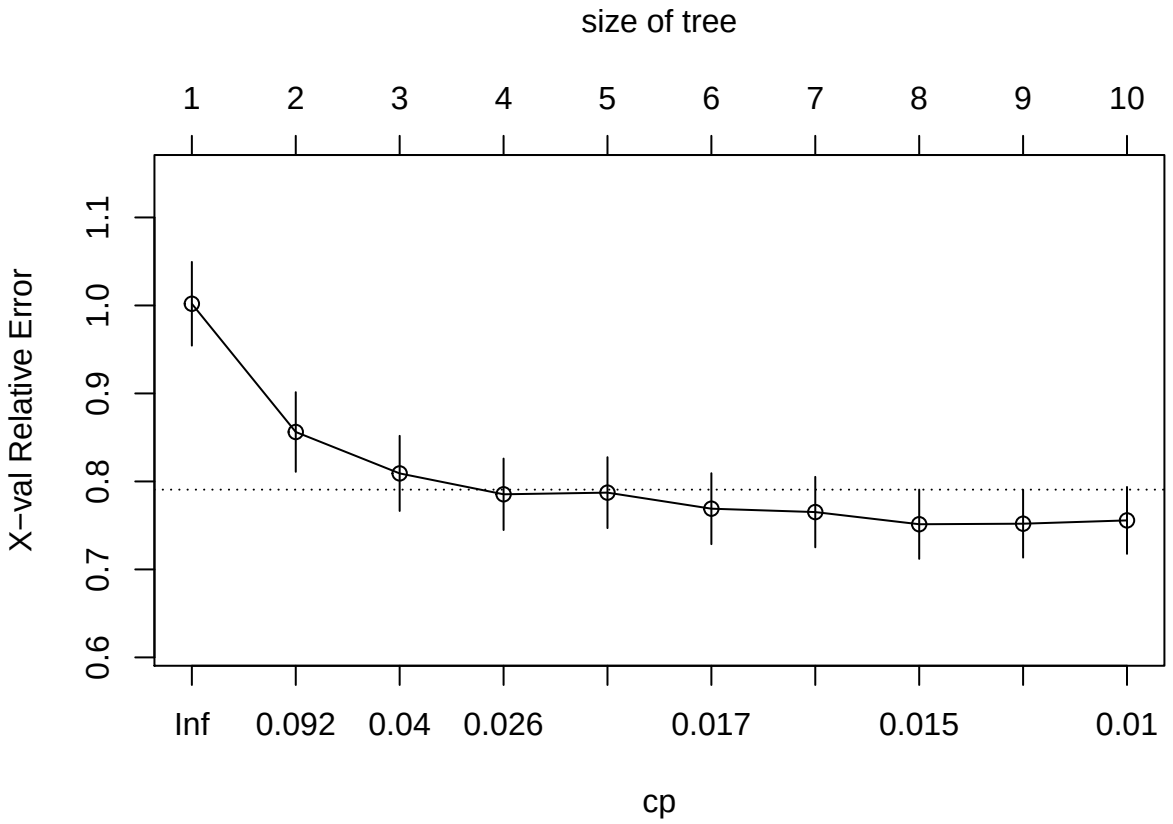
d)

```
printcp(t0)
```

```
##
## Regression tree:
## rpart(formula = quality ~ ., data = data, subset = train)
##
## Variables actually used in tree construction:
## [1] alcohol              sulphates            total.sulfur.dioxide
## [4] volatile.acidity
##
## Root node error: 673.61/1066 = 0.6319
##
## n= 1066
##
##          CP nsplit rel error  xerror     xstd
## 1  0.191933      0   1.00000 1.00191 0.047554
## 2  0.043796      1   0.80807 0.85619 0.045247
## 3  0.036949      2   0.76427 0.80911 0.042693
## 4  0.018752      3   0.72732 0.78537 0.040598
## 5  0.018514      4   0.70857 0.78725 0.040300
## 6  0.015369      5   0.69006 0.76901 0.040307
## 7  0.015315      6   0.67469 0.76516 0.040044
## 8  0.014166      7   0.65937 0.75134 0.039349
## 9  0.010301      8   0.64520 0.75196 0.038522
```

```
## 10 0.010000      9   0.63490 0.75571 0.037998
```

```
plotcp(t0)
```

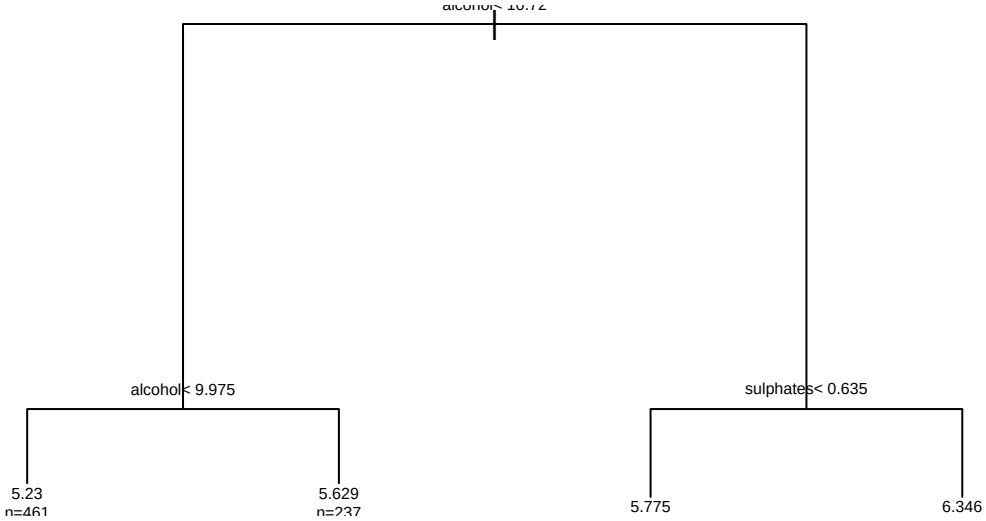


As shown in the above plot, cp = 0.026 is the optimal value for cp.

e)

We use the obtained cp for pruning our tree.

```
t1 <- prune(t0, 0.026)
plot(t1, branch=1, compress=TRUE)
text(t1, use.n = TRUE, cex=0.5)
```



The pruned tree is much simpler, only the variables *alcohol* and *sulphates* are used.

f)

we calculate the RMSE for the pruned model. The RMSE is slightli worse than for the unpruned tree.

```
sqrt(mean((data[test, 'quality'] - predict(t1, data[test,]))^2))
```

```
## [1] 0.7404386
```

g)

We use bagging to train 100 trees. For each tree we predict the test set and calculate the average prediction for each datapoint. We then canculate the RMSE usig the averaged predictions. We get our best results yet.

```
B <- 100
for (k in seq(1,B)) {
  tr <- sample(train, nrow(data), replace=TRUE)
  tree <- rpart(quality~., data, subset=tr)

  pred <- predict(tree, data[test,])
  if (! exists("pred_avg")) {
    pred_avg <- pred
  }
  pred_avg <- (pred + k*pred_avg) / (k+1)
}

sqrt(mean((data[test, 'quality'] - pred_avg )^2))
```

```
## [1] 0.6581207
```

h)

Here we calculate the MSE from OOB data instead from the test data.

```r
# function to merge arrays and calculate moving average
mergeavg <- function(src, new) {
  for (i in names(new)) {
    if (is.na(src[i,'v'])) {
      src[i, 'v'] <- new[i]
      src[i,'n'] <- 1
    } else {
      src[i, 'v'] <- (new[i] + src[i, 'n'] * src[i, 'v']) / (src[i, 'n'] + 1)
      src[i, 'n'] <- src[i, 'n'] + 1
    }
  }
  src
}


B <- 100

pred_avg <- array(numeric(0),dim=c(nrow(data), 2))
allidx <- seq(1,nrow(data))
dimnames(pred_avg) <- list(allidx, c('n','v'))

for (k in seq(1,B)) {
  tr <- sample(train, nrow(data), replace=TRUE)
  oob <- data[-tr,]
  tree <- rpart(quality~., data, subset=tr)

  pred <- predict(tree, oob)
  pred_avg <- mergeavg(pred_avg, pred)
}

mse_oob <- sum((data$quality - pred_avg[, 'v'])^2)/nrow(data)
mse_oob
```

```
## [1] 0.4152584
```

i)

we train random forest and evaluate the results. We can expect even better results since this algorithm restricts the choice of variables for each spit in the trees randomly, thus results in more diverse trees in the forest.

```r
#install.packages("randomForest")

library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```r
rf <- randomForest(quality~., data=data, subset=train, importance=TRUE)
pred <- predict(rf,data[test,])
sqrt(mean((data[test, 'quality'] -  pred)^2))
```

```
## [1] 0.6057203
```