# Decision Tree and Cross Validation

Suhail Shaikh

12/19/2019

**Q) This dataset is used to determine whether a wine quality is over 7. We have mapped the wine quality scores for you to binary classes of 0 and 1. Wine scores from 0 to 6 (inclusive) are mapped to 0, wine scores of 7 and above are mapped to 1. You will be performing binary classification on the dataset. The dataset is extracted from the UCI machine learning repository. Each line of this dataset describes a wine, using 12 columns: the firrst 11 describe the wines characteristics (details), and the last column is a ground truth label for the quality of the wine (0/1). Construct the best possible decision tree to predict the wine quality score. Explain how you have constructed your tree in details. Evaluate the performance of your decision tree using 10-fold cross validation. In a nutshell, you will first make a split of the provided data into 10 parts. Then hold out 1 part as the test set and use the remaining 9 parts for training. Train your decision tree using the training set and use the trained decision tree to classify entries in the test set. Repeat this process for all 10 parts, so that each entry will be used as the test set exactly once. To get the final accuracy value, take the average of the 10 folds accuracies (or other evaluation measures required).**

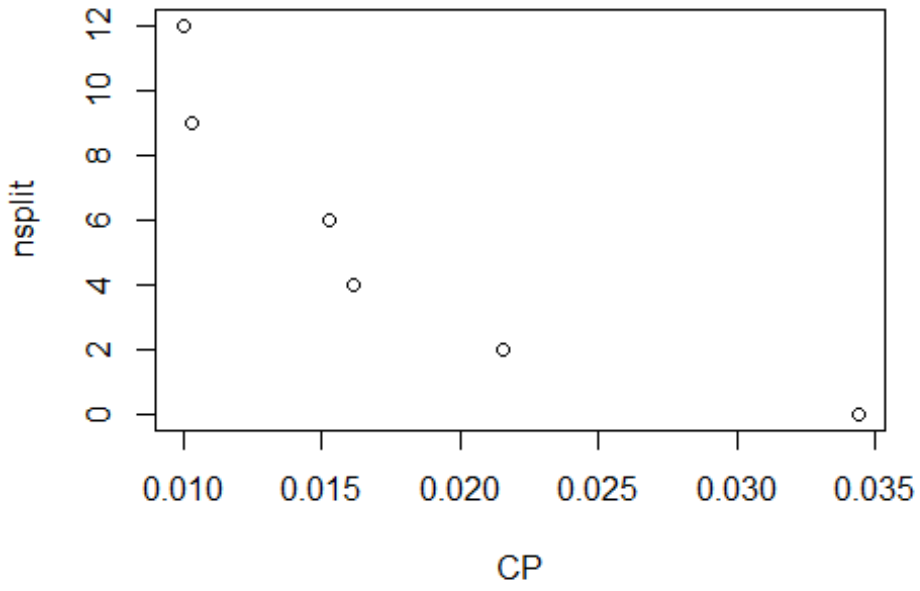
```
library(readxl)
```

```
library(readxl)
############################reading
file#########################################
wineData <- read.csv("wineData.csv")
str(wineData)
```

```
## 'data.frame':    4898 obs. of  12 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3
0.22 ...
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34
0.43 ...
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045
0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
```

```
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22
...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49
0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : int  0 0 0 0 0 0 0 0 0 0 ...

################################changing data type to factor for
classification problem#####################################
wineData$quality <- as.factor(wineData$quality)
summary(wineData)

##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0
##  Median :0.04300   Median : 34.00      Median :134.0
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0
##     density             pH          sulphates         alcohol
##  Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00
##  1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50
##  Median :0.9937   Median :3.180   Median :0.4700   Median :10.40
##  Mean   :0.9940   Mean   :3.188   Mean   :0.4898   Mean   :10.51
##  3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40
##  Max.   :1.0390   Max.   :3.820   Max.   :1.0800   Max.   :14.20
##  quality
##  0:3838
##  1:1060
##
##
##
##

#starting Cross validation
set.seed(1234)
################################installing
libraries#############################################################
######
library(MASS)

## Warning: package 'MASS' was built under R version 3.6.2
```

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(caTools)


library(caret)
# building a decision tree
library(caTools)
pointer <- sample.split(Y = wineData$quality, SplitRatio = .7)
train_dt <- wineData[pointer,]
test_dt <- wineData[!pointer,]

#model building
library(rpart)
mod_dt <- rpart(quality~., data = train_dt, method = "class", parms =
list(split="gini"))
plot(mod_dt$cptable)
```

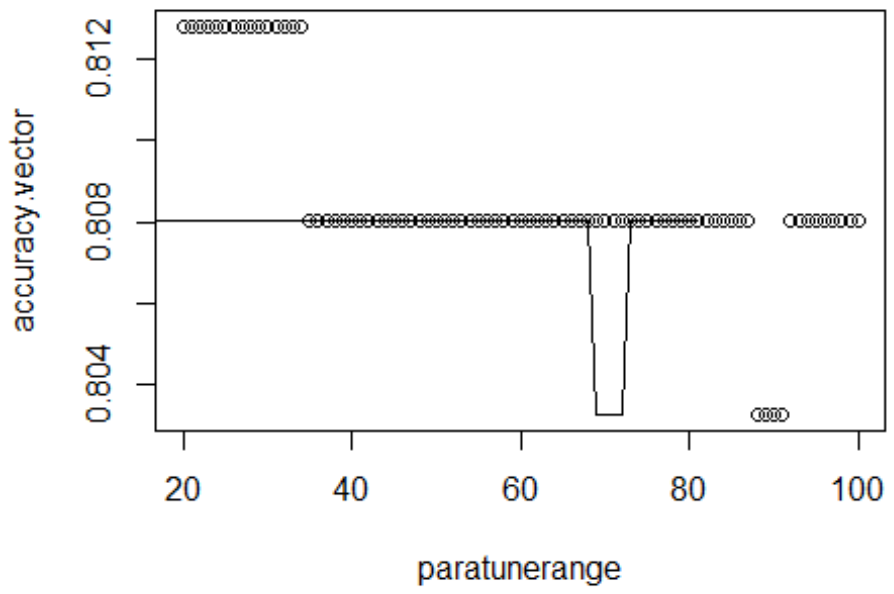



```
#there is not much of a difference when we use gini, information gain or
ctree to construct a tree, hence we are taking gini as a final variable.
#mod_dt$cptable expression list outs the cp values for the tree. We select
out the minimum CP value for which the xerror is minimum.
```

```r
# parameter tuning of decision tree,
# we are using two parameters to tune decision tree namely cp value and
minsplit. we have fixed cp value to .01(obtained from model) and running a
for loop
# for multiple time to obtain optimum minsplit value.
#function for accuracy prediction.............
accuracy_tune <- function(var){
  predict_withdt <- predict(var, test_dt, type = "class")
  frq <- table(predict_withdt, test_dt$quality)
  accuracy_val <- sum(diag(frq))/sum(frq)
  accuracy_val
}
###############for loop for determine accuracy for large range min
split##################

paratunerange <- c(20:100)
count<-1
accuracy.vector<-c()
for (i in paratunerange) {

mod_dt <- rpart(quality~., data = train_dt, method = "class", control =
rpart.control(minsplit = i,cp = 0.01))
accuracy.vector[count] <- accuracy_tune(mod_dt)
count<-count+1
}
accuracy.vector
```

```
##  [1] 0.8127978 0.8127978 0.8127978 0.8127978 0.8127978 0.8127978 0.8127978
##  [8] 0.8127978 0.8127978 0.8127978 0.8127978 0.8127978 0.8127978 0.8127978
## [15] 0.8127978 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327
## [22] 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327
## [29] 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327
## [36] 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327
## [43] 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327
## [50] 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327
## [57] 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327
## [64] 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327 0.8032675 0.8032675
## [71] 0.8032675 0.8032675 0.8080327 0.8080327 0.8080327 0.8080327 0.8080327
## [78] 0.8080327 0.8080327 0.8080327 0.8080327
```

```r
plot(paratunerange,accuracy.vector)
lines(accuracy.vector)
```




```
max <-max(accuracy.vector)
index = which(accuracy.vector==max)+19
index

##  [1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34

####### This suggest that this tree gives out maximum accuracy at minsplit
values of 20,21,22,23,24,25,26,27,28,29,30,31,32,33,34.
### max accuracy is 81.27%

######################Using Cross Validation Technique to determine
Accuracy#################################################33


winedata<-wineData[sample(nrow(wineData)),]
k <- 10
nmethod <- 1
folds <- cut(seq(1,nrow(winedata)),breaks=k,labels=FALSE)
models.err <- matrix(-1,k,nmethod, dimnames=list(paste0("Fold", 1:k),
c("rpart")))

for(i in 1:k)
{
  testIndexes <- which(folds==i, arr.ind=TRUE)
  test_DT <- winedata[testIndexes, ]
  train_DT <- winedata[-testIndexes, ]
```

```
  pntr <- sample(2, nrow(train_DT), replace = T, prob = c(0.7, 0.3))
  train_CV <- train_DT[pntr == 1, ]
  Validation_CV <- train_DT[pntr == 2, ]

  pr.err <- c()
    library(rpart)
    wine_rpart <- rpart(quality~., data = train_CV, method="class", control =
rpart.control(minsplit = 10, cp = 0.01))
    predicted <- predict(wine_rpart, newdata = Validation_CV, type = "class")
    pr.err <- c(pr.err,mean(Validation_CV$quality != predicted))
}
mean(pr.err)
```

```
## [1] 0.1888804
```

```
#This cross validation suggest that we have received 20% error from 10 fold
cross validation which is in tune with previous accuracy obtained with
decision tree.
```