Wine Quality Dataset

```
library(MASS)
library(dplyr)
library(tibble)
library(knitr)
library(readxl)
library(ROCR)
library(tidyr)
library(ggplot2)
library(statsr)
library(randomForest)
library(caret)
library(e1071)
library(pROC)
library(party)
library(rpart)
```

Problem 4:

We import the wine dataset from the csv file 'wineData.csv' and name our dataframe "wine". We are given that wine quality is a categorical variable with two classes. But presently it is a numerical variable as seen from the data summary. So we convert the variable "quality" into a factor variable.

```
wine <- read.csv("wineData.csv")
summary(wine)
```

```
##  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
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2164
##  3rd Qu.:0.0000
##  Max.   :1.0000
```

```
wine$quality <- factor(wine$quality)


colnames(wine)[colnames(wine)=="fixed.acidity"] <- "fixed_acidity"
colnames(wine)[colnames(wine)=="volatile.acidity"] <- "volatile_acidit
y"
colnames(wine)[colnames(wine)=="residual.sugar"] <- "residual_sugar"
colnames(wine)[colnames(wine)=="citric.acid"] <- "citric_acid"
colnames(wine)[colnames(wine)=="free.sulfur.dioxide"] <- "free_sulphur
"
colnames(wine)[colnames(wine)=="total.sulfur.dioxide"] <- "total_sulph
ur"
```

Splitting Training and Test sets

Let us split our data into training and test sets. We will split it in a 70:30 ratio, 70% in the training set and 30% in the test set. We also split the training data into training and validation sets.

```
set.seed(101)
wine_index <- sample.int(n = nrow(wine), size = floor(.70*nrow(wine)),
replace = F)
winetrain <- wine[wine_index,]
winetest <- wine[-wine_index,]


winetrain_index <- sample.int(n = nrow(wine),
                              size = floor(.70*nrow(winetrain)), repla
ce = F)
winetrtrain <- wine[winetrain_index,]
winetrval <- wine[-winetrain_index,]
```

We will first do parameter tuning with following steps:

1. First, we start by creating Model with default parameters
2. We will determine the best values of “cp” and “minsplit” parameters, to come to a more fine tuned model.
3. Evaluate the final model with chosen parameters on the test data using cross-validation.

Rpart decision tree using “gini”

We construct rpart decision tree using Gini index and find the best cp value using printcp().

```
library(rpart)

k=10
n = floor(nrow(winetrain)/k)
err.vect = rep(NA,k)

for (i in 1:k) {

  s1 = ((i-1) * n+1)
  s2 = (i*n)
  subset = s1:s2

  cvr.train = winetrain[-subset,]
  cvr.test = winetrain[subset,]

  winerpartgini <- rpart(quality ~ .-quality, data = cvr.train,
                         method = "class", parms = list(split = "gini" ))

  winepredgini_rpart <- prediction(
    predict(winerpartgini, newdata = cvr.test, type = "prob")[,2],
    cvr.test$quality)

  err.vect[i] <- performance(winepredgini_rpart, "auc")@y.values


  print(paste("AUC for fold", i, ":", err.vect[i]))

}

## [1] "AUC for fold 1 : 0.815129320210574"
## [1] "AUC for fold 2 : 0.773946260589366"
## [1] "AUC for fold 3 : 0.77221269296741"
## [1] "AUC for fold 4 : 0.657050465957122"
## [1] "AUC for fold 5 : 0.739284993949173"
## [1] "AUC for fold 6 : 0.791982323232323"
## [1] "AUC for fold 7 : 0.687401334216021"
## [1] "AUC for fold 8 : 0.676605387788359"
```

```
## [1] "AUC for fold 9 : 0.788862325242971"
## [1] "AUC for fold 10 : 0.736790966386555"
```

```r
print(paste("Average AUC :", mean(err.vect[[i]])))
```

```
## [1] "Average AUC : 0.736790966386555"
```

```r
printcp(winerpartgini)
```

```
##
## Classification tree:
## rpart(formula = quality ~ . - quality, data = cvr.train, method = "
class",
##     parms = list(split = "gini"))
##
## Variables actually used in tree construction:
## [1] alcohol         density         fixed_acidity   free_sulphur
## [5] pH              residual_sugar sulphates
##
## Root node error: 677/3086 = 0.21938
##
## n= 3086
##
##         CP nsplit rel error  xerror     xstd
## 1 0.039143      0   1.00000 1.00000 0.033957
## 2 0.018464      2   0.92171 0.95716 0.033421
## 3 0.014771      4   0.88479 0.96160 0.033478
## 4 0.010000      8   0.82422 0.92171 0.032957
```

The best value for rpart with gini, of cp comes out to be 0.01 and best value of minsplit is 8.

```r
set.seed(11123)

# Run the default model

ginituned <- rpart(quality ~ .-quality, data = winetrtrain,
                    method = "class", parms = list(split = "gini" ),
                    control = rpart.control(minsplit = 8, cp = 0.01)
)

ginicon <- predict(ginituned, winetrval, type = "class")

confusionMatrix(ginicon,  winetrval$quality)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
```

```
##          0 1844  408
##          1   90  157
##                                           
##                Accuracy : 0.8007          
##                  95% CI : (0.7845, 0.8162)
##     No Information Rate : 0.7739          
##     P-Value [Acc > NIR] : 0.0006352       
##                                           
##                   Kappa : 0.2889          
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
##                                           
##             Sensitivity : 0.9535          
##             Specificity : 0.2779          
##          Pos Pred Value : 0.8188          
##          Neg Pred Value : 0.6356          
##              Prevalence : 0.7739          
##          Detection Rate : 0.7379          
##    Detection Prevalence : 0.9012          
##       Balanced Accuracy : 0.6157          
##                                           
##        'Positive' Class : 0               
##
```

Rpart decision tree using "information gain"
We construct rpart decision tree using information gain index and find the best cp value using printcp().

```
k=10
n = floor(nrow(winetrain)/k)
err.vect = rep(NA,k)

for (i in 1:k) {

  s1 = ((i-1) * n+1)
  s2 = (i*n)
  subset = s1:s2

  cvr.train = winetrain[-subset,]
  cvr.test = winetrain[subset,]

  winerpartinfo <- rpart(quality ~ .-quality, data = cvr.train,
                       method = "class", parms = list(split = "informati
on" ))

  winepredinfo_rpart <- prediction(
```

```
    predict(winerpartinfo, newdata = cvr.test,
    type = "prob")[,2],
    cvr.test$quality )

  err.vect[i] <- performance(winepredinfo_rpart, "auc")@y.values


  print(paste("AUC for fold", i, ":", err.vect[i]))

}
```

```
## [1] "AUC for fold 1 : 0.793797207598993"
## [1] "AUC for fold 2 : 0.726105711761343"
## [1] "AUC for fold 3 : 0.739475618720902"
## [1] "AUC for fold 4 : 0.655089881346438"
## [1] "AUC for fold 5 : 0.719972771278741"
## [1] "AUC for fold 6 : 0.737519425019425"
## [1] "AUC for fold 7 : 0.691271579161786"
## [1] "AUC for fold 8 : 0.68948922951571"
## [1] "AUC for fold 9 : 0.761472896419105"
## [1] "AUC for fold 10 : 0.737106092436975"
```

```
print(paste("Average AUC :", mean(err.vect[[i]])))
```

```
## [1] "Average AUC : 0.737106092436975"
```

```
printcp(winerpartinfo)
```

```
##
## Classification tree:
## rpart(formula = quality ~ . - quality, data = cvr.train, method = "
class",
##     parms = list(split = "information"))
##
## Variables actually used in tree construction:
## [1] alcohol       chlorides     density       fixed_acidity free_su
lphur
## [6] pH
##
## Root node error: 677/3086 = 0.21938
##
## n= 3086
##
##          CP nsplit rel error  xerror     xstd
## 1 0.039143      0   1.00000 1.00000 0.033957
## 2 0.018464      2   0.92171 0.96160 0.033478
```

```
## 3 0.012186      4   0.88479 0.94535 0.033268
## 4 0.010000      8   0.83604 0.93501 0.033133
```

The best value for rpart with information gain, of cp comes out to be 0.01 and best value of minsplit is 8.

```
infotuned <- rpart(quality ~ .-quality, data = winetrtrain,
                    method = "class", parms = list(split = "informati
on" ),
                    control = rpart.control(minsplit = 8, cp = 0.01)
)

infocon <- predict(infotuned, winetrval, type = "class")

confusionMatrix(infocon,  winetrval$quality)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1844  408
##          1   90  157
## 
##                Accuracy : 0.8007
##                  95% CI : (0.7845, 0.8162)
##     No Information Rate : 0.7739
##     P-Value [Acc > NIR] : 0.0006352
## 
##                   Kappa : 0.2889
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
##             Sensitivity : 0.9535
##             Specificity : 0.2779
##          Pos Pred Value : 0.8188
##          Neg Pred Value : 0.6356
##              Prevalence : 0.7739
##          Detection Rate : 0.7379
##    Detection Prevalence : 0.9012
##       Balanced Accuracy : 0.6157
## 
##        'Positive' Class : 0
## 
```

The AUC for both Gini and information gain come to be the same. But the accuracy for Gini rpart is slightly higher than that of rpart information gain. So we will go ahead and check the performance of rpart using Gini on the test data

10-Fold Cross Validation for Model Evaluation on test data using Gini:

In a classification problem, we can measure the model performance using metrics like Area under the ROC curve or accuracy/error rate. We will use Area under the ROC curve as our measure of performance. To perform 10-fold cross validation, we have used a for loop which will run 10 times on different subsets of the “wine” data and generate a resulting Area under the ROC curve. The whole data is subsetted in 10 equal parts and each time the loop runs, one of those 10 is kept apart as the test set and remaining 9 subsets become a part of train set for that cross validation loop.

In the loop, we build the model using train and predict on test. We have used the prediction() and performance() functions, to calculate the area under curve for each loop(fold). After we get the AUC for all 10 folds, we take an average of them to find the overall performance of the model.

```
k=10
n = floor(nrow(wine)/k)
aucgini.vect = rep(NA,k)

for (i in 1:k) {

  s1 = ((i-1) * n+1)
  s2 = (i*n)
  subset = s1:s2

  cvrwine.train = wine[-subset,]
  cvrwine.test = wine[subset,]

  winerpartgini <- rpart(quality ~ .-quality, data = cvrwine.train,
                    method = "class", parms = list(split = "gini" ),
                    control = rpart.control(minsplit = 8, cp = 0.01))

  winepredgini_rpart <- prediction(
    predict(winerpartgini, newdata = cvrwine.test, type = "prob")[,2],
    cvrwine.test$quality )

  aucgini.vect[i] <- performance(winepredgini_rpart, "auc")@y.values


  print(paste("AUC for fold", i, ":", aucgini.vect[i]))

}

## [1] "AUC for fold 1 : 0.80751413950983"
## [1] "AUC for fold 2 : 0.692840223944875"
## [1] "AUC for fold 3 : 0.747063027649975"
```

```
## [1] "AUC for fold 4 : 0.800494653170927"
## [1] "AUC for fold 5 : 0.791452205882353"
## [1] "AUC for fold 6 : 0.830600970176069"
## [1] "AUC for fold 7 : 0.769527235354573"
## [1] "AUC for fold 8 : 0.757295907435183"
## [1] "AUC for fold 9 : 0.666986911723754"
## [1] "AUC for fold 10 : 0.730028488337587"
```

```r
print(paste("Average AUC :", mean(aucgini.vect[[i]])))
```

```
## [1] "Average AUC : 0.730028488337587"
```

```r
winetuned <- rpart(quality ~ .-quality, data = winetrain,
                    method = "class", parms = list(split = "gini" ),
                    control = rpart.control(minsplit = 8, cp = 0.01)
)

tunedcon <- predict(winetuned, winetest, type = "class")

confusionMatrix(tunedcon,  winetest$quality)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1108  221
##          1   49   92
##                                           
##                Accuracy : 0.8163          
##                  95% CI : (0.7956, 0.8358)
##     No Information Rate : 0.7871          
##     P-Value [Acc > NIR] : 0.002986        
##                                           
##                   Kappa : 0.3146          
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
##                                           
##             Sensitivity : 0.9576          
##             Specificity : 0.2939          
##          Pos Pred Value : 0.8337          
##          Neg Pred Value : 0.6525          
##              Prevalence : 0.7871          
##          Detection Rate : 0.7537          
##    Detection Prevalence : 0.9041          
##       Balanced Accuracy : 0.6258          
## 
```

```
##         'Positive' Class : 0
##
```

Finding:
We get an AUC of 0.737 on the training data and an AUC of 0.73 when performing cross validation on the test data.
The accuracy on training data came to be 0.8203 and that on test data it is 0.8163.