Red_AllModels_Final.R

nebojsahrnjez

2021-12-01

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v stringr 1.4.0
## v tidyr   1.1.4     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(moments)
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:purrr':
##
##     some

## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(ggplot2)
library(ggrepel)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
#Libraries from exploratory analysis

library(cvTools)
```

```
## Loading required package: lattice

## Loading required package: robustbase
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
#Libraries for this script

red <- read_csv("winequality-red.csv")
```

```
## Rows: 1599 Columns: 12

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
sum(is.na(red))
```

```
## [1] 0
```

```r
red <- na.omit(red)
#Reading in the data

dfr <- as.data.frame(red)

#Creating dataframe to be used

ALL.results <- data.frame(matrix(ncol=2,nrow=0,
                dimnames=list(NULL, c("Model", "Classification Accuracy %"))))

#Empty dataframe for results

set.seed(100)

test <- sample(1:nrow(dfr), size = nrow(dfr)/5)
train <- (-test)


dfr.train <- dfr[train,]
dfr.test <- dfr[test,]

#dfr train/test set

k <- 10 #number of folds

folds <- cvFolds(nrow(dfr), K=k)

#folds setup for cross-validation

r.mlm.train <- lm(
  quality ~.,
  data = dfr.train
)
#Create linear regression with all predictors using the training dataset

r.mlm.predict <- predict(r.mlm.train, newdata = dfr.test)
r.mlm.predict.rounded <- round(r.mlm.predict, digits = 0)

#Round the predicted to a integer so it can be compared to the test set for
# classification

(con_mat <- table(r.mlm.predict.rounded, dfr.test$quality))
```

```
##
## r.mlm.predict.rounded  3  4  5  6  7  8
##                     4  1  0  1  0  0  0
##                     5  3  2 98 36  0  0
##                     6  0  1 42 85 32  1
##                     7  0  0  1  5  9  2
```

```
r.mlm.acc <- round(mean(r.mlm.predict.rounded==dfr.test$quality)*100,digits = 2)

#Create the confusion matrix and calculate the proportion correct

ALL.results[1,] <- c("dfr MLR w/ Train/Test", r.mlm.acc)

#Record results in results matrix

r.mlm.cv.class <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

for(i in 1:k){
  tr.mlr <- dfr[folds$subsets[folds$which != i],]
  te.mlr <- dfr[folds$subsets[folds$which == i],]

  r.mlm <- lm(quality~., data = tr.mlr)
  r.mlm.pred <- predict(r.mlm, newdata = te.mlr)

  r.mlm.cv.class[i] <- mean(round(r.mlm.pred, digits = 0)==te.mlr$quality)
}

r.mlm.cv.class
```

```
##               1
##  [1,] 0.6312500
##  [2,] 0.6062500
##  [3,] 0.5875000
##  [4,] 0.5687500
##  [5,] 0.5437500
##  [6,] 0.6500000
##  [7,] 0.5812500
##  [8,] 0.5687500
##  [9,] 0.6187500
## [10,] 0.5849057
```

```
r.mlm.cv.class <- mean(r.mlm.cv.class)
print(paste("The average outputs correctly predicted is",
            round(r.mlm.cv.class*100,digits =2),"%",sep=" "))
```

```
## [1] "The average outputs correctly predicted is 59.41 %"
```

```
ALL.results[2,] <- c("dfr MLR w/ 10-fold CV", round(r.mlm.cv.class*100,
                                                    digits=2))

#dfr MLR with 10-fold cross-validation

dfr$quality <- factor(dfr$quality, ordered = TRUE)

dfr.train <- dfr[train,]
dfr.test <- dfr[test,]

r.olr <- polr(quality~., data = dfr.train, Hess = TRUE)
```

```
r.olr.pred <- predict(r.olr, newdata = dfr.test)

r.olr.pred <- as.numeric(as.character(unlist(r.olr.pred)))
dfr.test$quality <- as.numeric(as.character(unlist(dfr.test$quality)))

r.olr.class <- mean(r.olr.pred == dfr.test$quality)

ALL.results[3,] <- c("dfr OLR w/ Training/Test", round(r.olr.class*100,
                                                       digits=2))

#dfr OLR with training/test set

r.olr.cv.class <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

for(i in 1:k){
  tr.olr <- dfr[folds$subsets[folds$which != i],]
  te.olr <- dfr[folds$subsets[folds$which == i],]

  r.olr.cv <- polr(quality~., data = tr.olr, Hess = TRUE)
  r.olr.cv.pred <- predict(r.olr.cv, newdata = te.olr)

  r.olr.cv.pred <- as.numeric(as.character(unlist(r.olr.cv.pred)))
  te.olr$quality <- as.numeric(as.character(unlist(te.olr$quality)))

  r.olr.cv.class[i] <- mean(r.olr.cv.pred==te.olr$quality)
}

r.olr.cv.class
```

```
##               1
##  [1,] 0.6250000
##  [2,] 0.6062500
##  [3,] 0.5750000
##  [4,] 0.5750000
##  [5,] 0.5125000
##  [6,] 0.6500000
##  [7,] 0.5562500
##  [8,] 0.5562500
##  [9,] 0.6312500
## [10,] 0.5974843
```

```
r.olr.cv.class <- mean(r.olr.cv.class)
print(paste("The average outputs correctly predicted is",
            round(r.olr.cv.class*100,digits =2),"%",sep=" "))
```

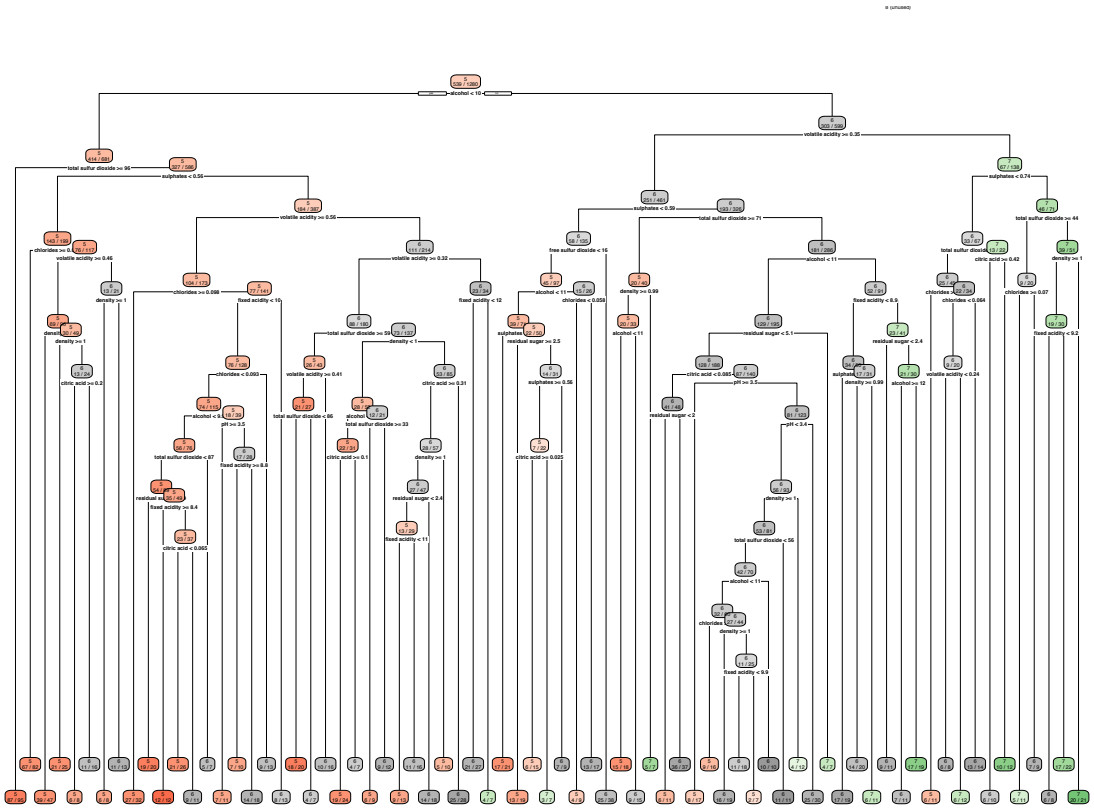
```
## [1] "The average outputs correctly predicted is 58.85 %"
```

```
ALL.results[4,] <- c("dfr OLR w/ 10-fold CV", round(r.olr.cv.class*100,
                                                    digits=2))

#dfr OLR with 10-fold cross validation

ALL.results
```

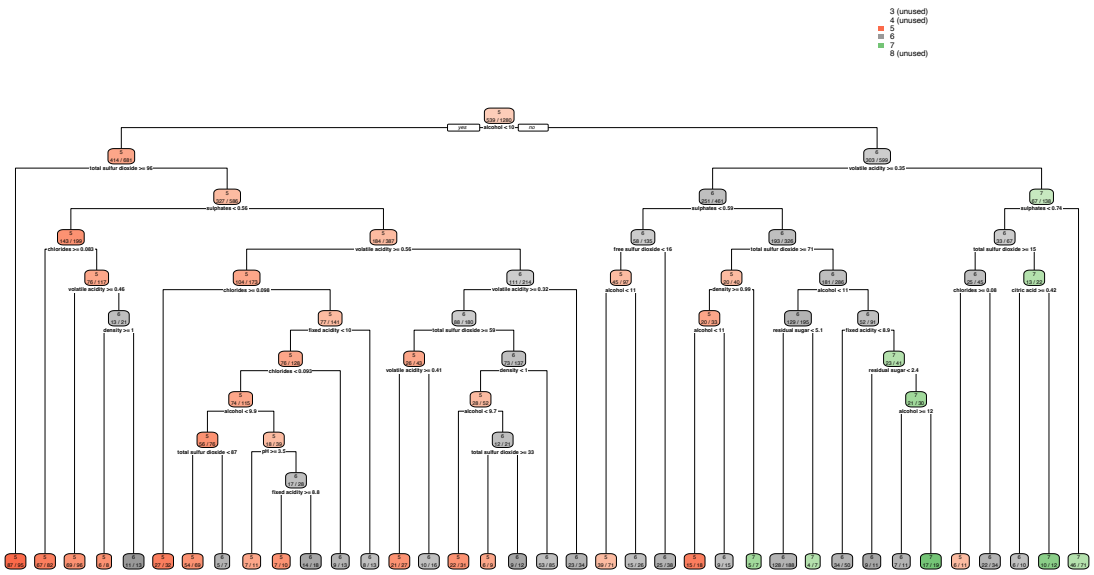
```
##                       Model Classification.Accuracy..
## 1    dfr MLR w/ Train/Test                       60.19
## 2    dfr MLR w/ 10-fold CV                       59.41
## 3 dfr OLR w/ Training/Test                       60.82
## 4    dfr OLR w/ 10-fold CV                       58.85
```

```
### END OF REGRESSION SECTION ###

library(cvTools)
library(rpart)
library(rpart.plot)
library(rpartScore)
#Libraries for this section

set.seed(100)

test <- sample(1:nrow(dfr), size = nrow(dfr)/5)
train <- (-test)

dfr <- as.data.frame(red)

dfr.train <- dfr[train,]
dfr.test <- dfr[test,]

#Quickly resetting the dataframes

r.tree <- rpart(quality~., data = dfr.train, method = "class", cp = 0.000001)

r.tree.pred <- predict(r.tree, newdata = dfr.test, type = "class")

r.tree.class <- mean(r.tree.pred == dfr.test$quality)

ALL.results[5,] <- c("dfr CART w/ Train/Test", round(r.tree.class*100,
                                                     digits = 2))

#dfr simple Classifcation Tree

rpart.plot(r.tree, extra = 2, digits = 2)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
#plot dfr simple classification tree

cp <- data.frame(r.tree$cptable)
min.cp <- which.min(cp$xerror)
cp <- cp$CP[min.cp]

#Find cp with minimum relative error for dfr classification tree

r.tree.prun <- prune(r.tree, cp = cp)

r.tree.pred <- predict(r.tree.prun, newdata = dfr.test, type = "class")

r.tree.class <- mean(r.tree.pred == dfr.test$quality)

ALL.results[6,] <- c("dfr pruned CART w/ Train/Test", round(r.tree.class*100,
                                                            digits = 2))

rpart.plot(r.tree.prun, extra = 2, digits = 2)
```




```
#Plot pruned dfr classification tree

k <- 10 #number of folds

folds <- cvFolds(nrow(dfr), K=k)

r.tree.cv.class <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

#Setting up for cross-fold validation

for(i in 1:k){
  tr.tree <- dfr[folds$subsets[folds$which != i],]
  te.tree <- dfr[folds$subsets[folds$which == i],]

  r.tree.cv <- rpart(quality~., data = tr.tree, method = "class",
                     cp = 0.000001)

  cp.cv <- data.frame(r.tree.cv$cptable)
  min.cp.cv <- which.min(cp.cv$xerror)
  cp.cv <- cp.cv$CP[min.cp.cv]

  r.tree.prun.cv <- prune(r.tree.cv, cp = cp.cv)

  r.tree.pred.cv <- predict(r.tree.prun.cv, newdata = te.tree, type = "class")

  r.tree.cv.class[i] <- mean(r.tree.pred.cv == te.tree$quality)
```

```
}

r.tree.cv.class
```

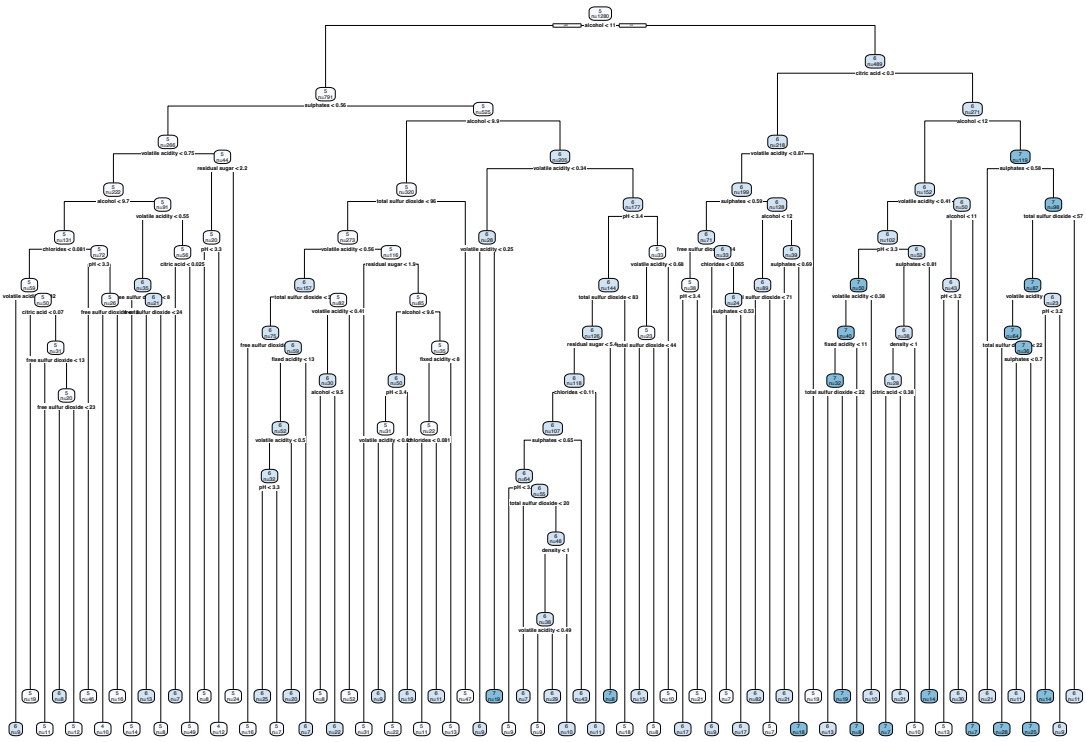
```
##               1
##  [1,] 0.6000000
##  [2,] 0.5875000
##  [3,] 0.5625000
##  [4,] 0.6375000
##  [5,] 0.5562500
##  [6,] 0.6062500
##  [7,] 0.5937500
##  [8,] 0.6250000
##  [9,] 0.6125000
## [10,] 0.5974843
```

```
r.tree.cv.class <- mean(r.tree.cv.class)
print(paste("The average outputs correctly predicted is",
            round(r.tree.cv.class*100,digits =2),"%",sep=" "))
```

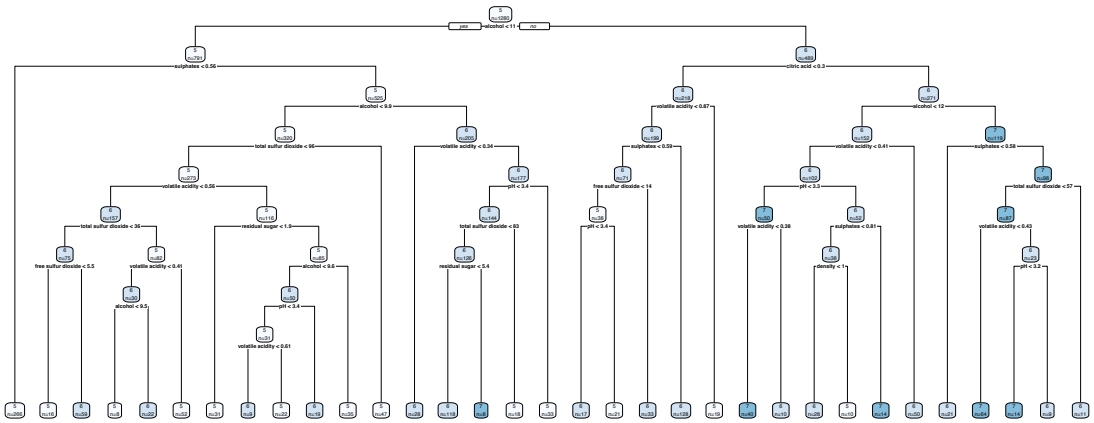
```
## [1] "The average outputs correctly predicted is 59.79 %"
```

```
ALL.results[7,] <- c("dfr CART w/ 10-fold CV", round(r.tree.cv.class*100,
                                                     digits=2))

#dfr CART with 10-fold cross validation

r.ordtree <- rpartScore(quality~., data = dfr.train,prune = "mr",cp= 0.000001)

r.ordtree.pred <- predict(r.ordtree, newdata = dfr.test)

r.ordtree.class <- mean(r.ordtree.pred == dfr.test$quality)

ALL.results[8,] <- c("dfr Ordinal Tree w/ Train/Test",
                     round(r.ordtree.class*100,digits = 2))

rpart.plot(r.ordtree, extra = 1, digits = 2)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
#dfw simple ordinal tree with training/test set

ordcp <- data.frame(r.ordtree$cptable)
min.ordcp <- which.min(ordcp$xerror)
ordcp <- ordcp$CP[min.ordcp]

#Find cp with minimum relative error for dfw ordinal tree

r.ordtree.prun <- prune(r.ordtree, cp = ordcp)

r.ordtree.pred <- predict(r.ordtree.prun, newdata = dfr.test)

r.ordtree.class <- mean(r.ordtree.pred == dfr.test$quality)

ALL.results[9,] <- c("dfr pruned Ordinal Tree w/ Train/Test",
                     round(r.ordtree.class*100,digits = 2))

rpart.plot(r.ordtree.prun, extra = 1, digits = 2)
```



```
#dfr pruned ordinal tree with training/test set

r.ordtree.class.cv <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

for(i in 1:k){
  tr.ord <- dfr[folds$subsets[folds$which != i],]
  te.ord <- dfr[folds$subsets[folds$which == i],]

  r.ordtree.cv <- rpartScore(quality~., data = tr.ord,prune = "mr",
                             cp=0.000001)

  ordcp.cv <- data.frame(r.ordtree.cv$cptable)
  min.ordcp.cv <- which.min(ordcp.cv$xerror)
  ordcp.cv <- ordcp.cv$CP[min.ordcp.cv]

  r.ordtree.prun.cv <- prune(r.ordtree.cv, cp = ordcp.cv)

  r.ordtree.pred.cv <- predict(r.ordtree.prun.cv, newdata = te.ord)

  r.ordtree.class.cv[i] <- mean(r.ordtree.pred.cv == te.ord$quality)

}

r.ordtree.class.cv
```

```
##               1
```

```
##  [1,] 0.6250000
##  [2,] 0.5812500
##  [3,] 0.5812500
##  [4,] 0.6812500
##  [5,] 0.5375000
##  [6,] 0.5937500
##  [7,] 0.6250000
##  [8,] 0.6812500
##  [9,] 0.6125000
## [10,] 0.6540881
```

```
r.ordtree.class.cv <- mean(r.ordtree.class.cv)
print(paste("The average outputs correctly predicted is",
            round(r.ordtree.class.cv*100,digits =2),"%",sep=" "))
```

```
## [1] "The average outputs correctly predicted is 61.73 %"
```

```
ALL.results[10,] <- c("dfr pruned ordinal tree w/ 10-fold CV",
                      round(r.ordtree.class.cv*100,digits=2))

#dfr pruned ordinal tree with cross-validation

ALL.results
```

```
##                                  Model Classification.Accuracy..
## 1                 dfr MLR w/ Train/Test                    60.19
## 2                 dfr MLR w/ 10-fold CV                    59.41
## 3               dfr OLR w/ Training/Test                   60.82
## 4                  dfr OLR w/ 10-fold CV                   58.85
## 5                dfr CART w/ Train/Test                    56.11
## 6          dfr pruned CART w/ Train/Test                   56.74
## 7                dfr CART w/ 10-fold CV                    59.79
## 8         dfr Ordinal Tree w/ Train/Test                   57.68
## 9  dfr pruned Ordinal Tree w/ Train/Test                    60.5
## 10 dfr pruned ordinal tree w/ 10-fold CV                   61.73
```

```
### END OF TREE SECTION ###

library(cvTools)
library(randomForest)
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

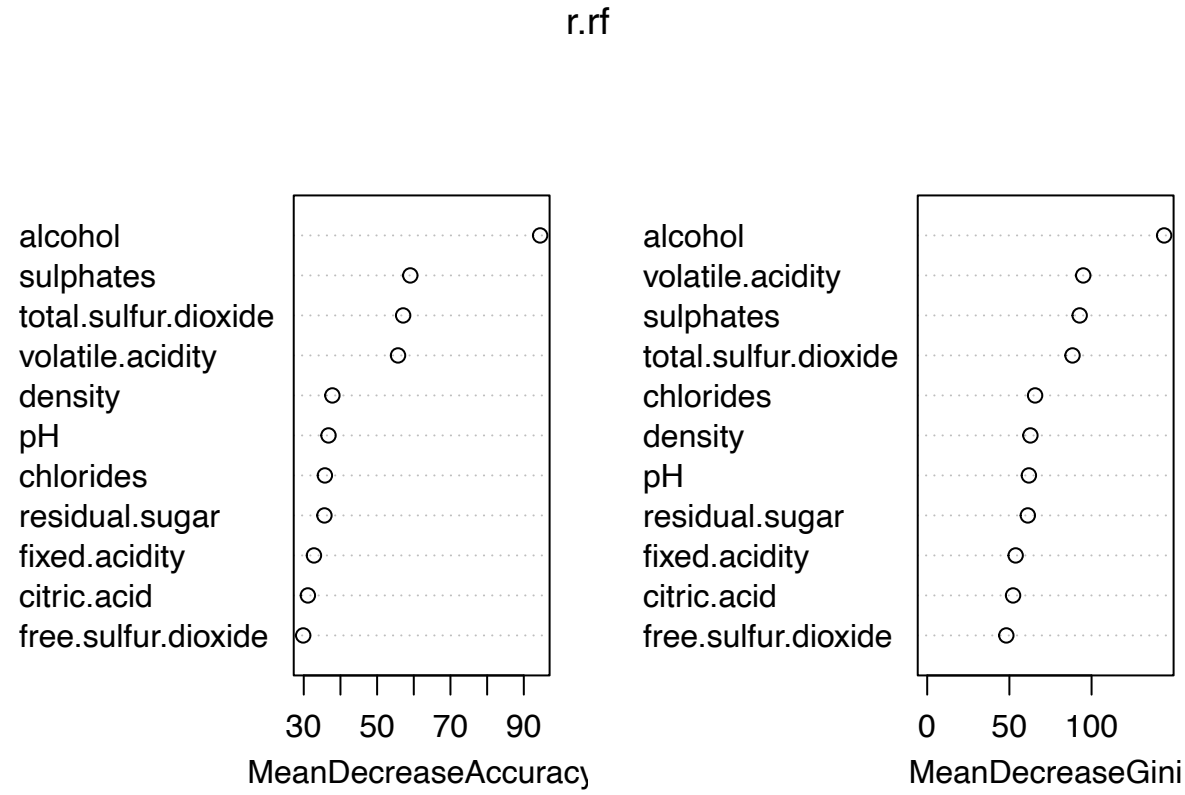
```
## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ordinalForest)
#Libraries for this section

set.seed(100)

test <- sample(1:nrow(dfr), size = nrow(dfr)/5)
train <- (-test)

dfr <- as.data.frame(red)

dfr$quality <- as.factor(dfr$quality)
names(dfr) <- make.names(names(dfr))

dfr.train <- dfr[train,]
dfr.test <- dfr[test,]

#Quickly resetting the dataframes

r.rf <- randomForest(quality~., data = dfr.train, mtry =11, ntree = 500,
                     importance = TRUE)

r.rf.pred <- predict(r.rf, newdata = dfr.test)

r.rf.class <- mean(r.rf.pred == dfr.test$quality)

ALL.results[11,] <- c("dfr random forest w/ training/test set",
                    round(r.rf.class*100,digits=2))

importance(r.rf)
```

```
##                              3          4        5        6        7         8
## fixed.acidity         1.8445339 -3.0405308 21.95349 16.86892 20.02870  3.020042
## volatile.acidity      0.1324556 11.3348347 41.06736 20.12439 47.04340 11.552658
## citric.acid          -1.0010015  0.3540928 19.96609 15.90889 16.82706  5.811584
## residual.sugar       -1.3440623  1.4559542 25.80756 16.83932 23.15948  3.485665
## chlorides            -1.7514236 -2.4940402 31.61493 17.53044 13.04937  3.667758
## free.sulfur.dioxide   0.0000000  1.6183842 19.99990 17.96019 15.11291  3.827889
## total.sulfur.dioxide -0.6327087  2.3965713 36.96368 29.68338 29.88857  6.263533
## density               0.7279924 -0.8366595 24.59042 22.58646 18.04341  7.210200
## pH                   -1.0010015  2.9375897 27.34432 15.98300 17.76732  6.260108
## sulphates            -1.1783295  7.7943374 37.65956 25.46382 43.98427 10.122259
## alcohol               0.6327087  2.3604041 74.58519 31.77753 62.96021 12.627484
##                      MeanDecreaseAccuracy MeanDecreaseGini
## fixed.acidity                    32.77575         53.80926
## volatile.acidity                 55.69854         94.93061
```

```
## citric.acid                31.11879         52.23405
## residual.sugar             35.63008         61.21456
## chlorides                  35.75297         65.64648
## free.sulfur.dioxide        29.84596         48.13933
## total.sulfur.dioxide       57.09615         88.40633
## density                    37.80009         62.78252
## pH                         36.74176         61.85535
## sulphates                  59.04827         92.76501
## alcohol                    94.44849        144.14195
```

```
varImpPlot(r.rf)
```




```
#dwr random forest with training/test set

k <- 10 #number of folds

folds <- cvFolds(nrow(dfr), K=k)

r.rf.cv.class <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

for(i in 1:k){
  tr.rf <- dfr[folds$subsets[folds$which != i],]
  te.rf <- dfr[folds$subsets[folds$which == i],]

  r.rf.cv <- randomForest(quality~., data = tr.rf, mtry =11, ntree = 500,
```

```
                          importance = TRUE)

  r.rf.pred.cv <- predict(r.rf.cv, newdata = te.rf)

  r.rf.cv.class[i] <- mean(r.rf.pred.cv == te.rf$quality)

}

r.rf.cv.class
```

```
##               1
##  [1,] 0.6812500
##  [2,] 0.6250000
##  [3,] 0.7187500
##  [4,] 0.7500000
##  [5,] 0.7125000
##  [6,] 0.7125000
##  [7,] 0.6687500
##  [8,] 0.7437500
##  [9,] 0.7250000
## [10,] 0.6666667
```

```
r.rf.cv.class <- mean(r.rf.cv.class)
print(paste("The average outputs correctly predicted is",
            round(r.rf.cv.class*100,digits =2),"%",sep=" "))
```

```
## [1] "The average outputs correctly predicted is 70.04 %"
```

```
ALL.results[12,] <- c("dfr Random Forest w/ 10-fold CV",
                      round(r.rf.cv.class*100,digits=2))

#dfr random forest with cross-validation

r.ordfor <- ordfor("quality", data = dfr.train, mtry = 11,
                   nsets = 100, ntreeperdiv = 10, ntreefinal = 500,
                   nbest = 1, npermtrial = 50)
```

```
## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf
```

```
## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf
```

```
## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf
```

```
## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf
```

```
## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf
```

```
## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf
```

```r
r.ordfor.pred <- predict(r.ordfor, newdata = dfr.test)

r.ordfor.class <- mean(r.ordfor.pred$ypred==dfr.test$quality)

ALL.results[13,] <- c("dfr Ordinal Forest w/ training/test set",
                      round(r.ordfor.class*100,digits=2))
```

```r
head(sort(r.ordfor$varimp, decreasing = TRUE), 4)
```

```
##               alcohol            sulphates     volatile.acidity
##            0.04154180           0.02442940           0.02121745
## total.sulfur.dioxide
##            0.01499230
```

```r
#dfr ordinal forest with training/test set

r.ordfor.cv.class <- matrix(NA,k,1, dimnames=list(NULL, paste(1)))

for(i in 1:k){
  tr.of <- dfr[folds$subsets[folds$which != i],]
  te.of <- dfr[folds$subsets[folds$which == i],]

  r.ordfor.cv <- ordfor("quality", data = tr.of, mtry = 11,
                        nsets = 100, ntreeperdiv = 10, ntreefinal = 500,
                        nbest = 1, npermtrial = 50)

  r.ordfor.pred.cv <- predict(r.ordfor.cv, newdata = te.of)

  r.ordfor.cv.class[i] <- mean(r.ordfor.pred.cv$ypred == te.of$quality)

}
```

```
## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
```

```
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf

## Warning in max(which(x >= qnorm(bordersb[b, ]))): no non-missing arguments to
## max; returning -Inf
```

```
r.ordfor.cv.class
```

```
##              1
##  [1,] 0.637500
##  [2,] 0.612500
##  [3,] 0.700000
##  [4,] 0.750000
##  [5,] 0.700000
##  [6,] 0.737500
##  [7,] 0.700000
##  [8,] 0.731250
##  [9,] 0.725000
## [10,] 0.672956
```

```
r.ordfor.cv.class <- mean(r.ordfor.cv.class)
print(paste("The average outputs correctly predicted is",
            round(r.ordfor.cv.class*100,digits =2),"%",sep=" "))
```

```
## [1] "The average outputs correctly predicted is 69.67 %"
```

```
ALL.results[14,] <- c("dfw Ordinal Forest w/ 10-fold CV",
                      round(r.ordfor.cv.class*100,digits=2))

#dfr ordinal forest with 10-fold cross validation

ALL.results
```

```
##                                    Model Classification.Accuracy..
## 1                     dfr MLR w/ Train/Test                     60.19
```

```
## 2                    dfr MLR w/ 10-fold CV                    59.41
## 3                 dfr OLR w/ Training/Test                    60.82
## 4                    dfr OLR w/ 10-fold CV                    58.85
## 5                   dfr CART w/ Train/Test                    56.11
## 6             dfr pruned CART w/ Train/Test                    56.74
## 7                    dfr CART w/ 10-fold CV                    59.79
## 8           dfr Ordinal Tree w/ Train/Test                    57.68
## 9    dfr pruned Ordinal Tree w/ Train/Test                     60.5
## 10   dfr pruned ordinal tree w/ 10-fold CV                    61.73
## 11  dfr random forest w/ training/test set                    68.97
## 12         dfr Random Forest w/ 10-fold CV                    70.04
## 13 dfr Ordinal Forest w/ training/test set                    68.03
## 14        dfw Ordinal Forest w/ 10-fold CV                    69.67
```