RFWineQuality.R

ai

Mon Jun 5 18:20:00 2017

```
# Reference for data source (
# @misc{Lichman:2013 ,
# author = "M. Lichman",
# year = "2013",
# title = "{UCI} Machine Learning Repository",
# url = "http://archive.ics.uci.edu/ml",
# institution = "University of California, Irvine, School of Information and Computer Sciences" })

# Decision Trees
# Source of Data Set:- UCI Repository - Wine Quality Data(https://archive.ics.uci.edu/ml/datasets/wine+

# Exploring and preparing the data
# Step 2: Exploring and preparing the data
# Read the csv file into a data frame titled WineData.
WineData <- read.table("winequality-red.csv", sep=";", header=TRUE)

# Creating a categorical variable for wine quality
# WineData$quality <- ifelse(WineData$quality == 3, "Lev_Three", ifelse(WineData$quality == 4, "Lev_Fou
# WineData$quality <- as.factor(WineData$quality)


WineData$quality <- ifelse(WineData$quality < 5, 'bad', ifelse(WineData$quality > 6,'good','normal'))
WineData$quality <- as.factor(WineData$quality)
str(WineData$quality)
```

```
##  Factor w/ 3 levels "bad","good","normal": 3 3 3 3 3 3 3 2 2 3 ...
```

```
# Data preparation - creating random training and test datasets
# Create random sample
# Divide the data into a training set and a test set randomly with ratio 80:20

set.seed(123)
train_sample <- sample(nrow(WineData), 0.8 * nrow(WineData))
WineData_train <- WineData[train_sample, ]
WineData_test <- WineData[-train_sample, ]

# Check whether data set fairly even split
prop.table(table(WineData_train$quality))
```

```
##
##        bad       good     normal
## 0.03909304 0.13995309 0.82095387
```

```
prop.table(table(WineData_test$quality))
```

```
##
##      bad     good   normal
## 0.040625 0.118750 0.840625
```

```
## Training random forests
library(randomForest)
```

```
## randomForest 4.6-12
```

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

```
# set.seed(300)
rf_model <- randomForest(WineData_train[-12], WineData_train$quality)
rf_model
```

```
##
## Call:
##  randomForest(x = WineData_train[-12], y = WineData_train$quality)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.76%
## Confusion matrix:
##        bad good normal class.error
## bad      0    0     50  1.00000000
## good     0   86     93  0.51955307
## normal   1   32   1017  0.03142857
```

```
# Evaluating random forest performance
# Making predictions
rf_predict <- predict(rf_model, WineData_test)

# Various R Programming Tools for Model Fitting
library(gmodels)

# create a cross tabulation indicating the agreement between the two vectors.
# Specifying prop.chisq = FALSE will remove the unnecessary chi-square
# values from the output.
# Setting the prop.c and prop.r parameters to FALSE removes the column and row percentages
# from the table. The remaining percentage ( prop.t ) indicates the proportion of
# records in the cell out of the total number of records:
CrossTable(WineData_test$quality, rf_predict, prop.chisq = FALSE, prop.c= FALSE, prop.r = FALSE, dnn = c
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  320
##
##
##                | Predicted quality
## Actual quality |      good |    normal | Row Total |
## ---------------|-----------|-----------|-----------|
##            bad |         0 |        13 |        13 |
```

```
##                 |     0.000 |     0.041 |           |
## ----------------|-----------|-----------|-----------|
##            good |        22 |        16 |        38 |
##                 |     0.069 |     0.050 |           |
## ----------------|-----------|-----------|-----------|
##          normal |         5 |       264 |       269 |
##                 |     0.016 |     0.825 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |        27 |       293 |       320 |
## ----------------|-----------|-----------|-----------|
##
##
```

```
# Accuracy : Measures of performance
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

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

```
confusionMatrix(WineData_test$quality, rf_predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good normal
##     bad      0    0     13
##     good     0   22     16
##     normal   0    5    264
##
## Overall Statistics
##
##                Accuracy : 0.8938
##                  95% CI : (0.8547, 0.9253)
##     No Information Rate : 0.9156
##     P-Value [Acc > NIR] : 0.9303
##
##                   Kappa : 0.5177
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: bad Class: good Class: normal
## Sensitivity                  NA     0.81481        0.9010
## Specificity             0.95937     0.94539        0.8148
## Pos Pred Value               NA     0.57895        0.9814
## Neg Pred Value               NA     0.98227        0.4314
## Prevalence              0.00000     0.08438        0.9156
## Detection Rate          0.00000     0.06875        0.8250
## Detection Prevalence    0.04063     0.11875        0.8406
```

```
## Balanced Accuracy            NA     0.88010        0.8579
```

```r
# Evaluating random forest performance
# use training control option and use repeated 10-fold cross-validation or 10-fold CV repeated 10 times
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 10)

# Set up the tuning grid for the random forest
grid_rf <- expand.grid(.mtry = c(2, 4, 8, 16))

# Use kappa metric to select the best model
set.seed(300)
m_rf <- train(quality ~ ., data = WineData, method = "rf", metric = "Kappa", trControl= ctrl, tuneGrid =
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid
## mtry: reset to within valid range
```

```r
# Compare the boosted tree using 10,20,30 and 40 iterations

grid_c50 <- expand.grid(.model = "tree", .trials = c(10, 20, 30, 40),.winnow = "FALSE")

set.seed(300)
m_c50 <- train(quality ~ ., data = WineData, method = "C5.0", metric = "Kappa", trControl= ctrl, tuneGr
```

```
## Loading required package: C50
```

```
## Loading required package: plyr
```

```
## Warning in Ops.factor(x$winnow): '!' not meaningful for factors
```

```r
# Compare the two approaches
m_rf
```

```
## Random Forest
##
## 1599 samples
##   11 predictors
##    3 classes: 'bad', 'good', 'normal'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 1439, 1440, 1440, 1439, 1439, 1439, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8751889  0.4721643
##    4    0.8759369  0.4888957
##    8    0.8741142  0.4958287
##   16    0.8727357  0.4952184
##
## Kappa was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 8.
```

```r
m_c50
```

```
## C5.0
##
## 1599 samples
##   11 predictors
##    3 classes: 'bad', 'good', 'normal'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 1439, 1440, 1440, 1439, 1439, 1439, ...
## Resampling results across tuning parameters:
##
##   trials  Accuracy   Kappa
##   10      0.8636042  0.4574296
##   20      0.8672996  0.4715870
##   30      0.8689876  0.4789028
##   40      0.8689911  0.4761948
##
## Tuning parameter 'model' was held constant at a value of tree
##
## Tuning parameter 'winnow' was held constant at a value of FALSE
## Kappa was used to select the optimal model using  the largest value.
## The final values used for the model were trials = 30, model = tree
##  and winnow = FALSE.
```