Predicting Wine Quality

Shaheen Gauher, Data Scientist at Microsoft

Problem: Predict wine quality using Multiclass Classification analysis.The data contains quality ratings for a few thousands of wines (1599 red wine samples), along with their physical and chemical properties (11 predictors). We want to use these properties to predict a rating for a wine. The goal is to model wine quality based on physicochemical tests.

(**data from** https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv (https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv))
Azure ML experiment at https://gallery.cortanaintelligence.com/Experiment/Predict-Wine-Quality-Classification-10 (https://gallery.cortanaintelligence.com/Experiment/Predict-Wine-Quality-Classification-10)

In [79]:
```
rm(list=ls())
#download data from
#https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.dat
a
data_wine = read.table("https://archive.ics.uci.edu/ml/machine-learning-
databases/wine-quality/winequality-red.csv",
                       header=T,sep=";",na.strings="NA") #1599 12
class(data_wine)  #"data.frame"

#Relabel quality ratings as follows
#3,4,5 as Low
#6 as Med
#7,8 as High

data_wine$qualityV2 = ifelse(data_wine$quality <=5, 'Low','None')
TorF = data_wine$quality == 6
data_wine$qualityV2[TorF] = 'Med'
TorF = data_wine$quality > 6
data_wine$qualityV2[TorF] = 'High'

data_wine$quality = NULL
names(data_wine)[names(data_wine)=='qualityV2'] = 'quality'

#convert data frame to xdf object using rxDataStep()
pathc     = getwd()
dataclassi_xdf = file.path(pathc,'dataclassi_xdf.xdf')
data_classi = rxDataStep(inData = data_wine, outFile = dataclassi_xdf ,
                       rowsPerRead=500, overwrite=TRUE, reportProgress=
0)
class(data_classi)  # "RxXdfData"
```

Out[79]: "data.frame"

Out[79]: "RxXdfData"

In [78]:
```
#make a new column factorQuality from quality col -- make it categorical
rxFactors(inData = data_classi, outFile = data_classi, overwrite = TRUE,
          factorInfo = list(factorQuality = list(varName = "quality")),r
eportProgress=0)

#can remove the col quality now
ColsToKeep = setdiff(names(data_classi),c('quality'))

data_classi = rxDataStep(inData = data_classi, outFile = 'data_classi_te
mp.xdf',varsToKeep = ColsToKeep, overwrite = TRUE)

#rename the label col 'factorQuality' as 'LabelsCol'
names(data_classi)[names(data_classi)=='factorQuality'] = 'LabelsCol'
```

```
Rows Read: 500, Total Rows Processed: 500, Total Chunk Time: 0.015 se
conds
Rows Read: 500, Total Rows Processed: 1000, Total Chunk Time: Less th
an .001 seconds
Rows Read: 500, Total Rows Processed: 1500, Total Chunk Time: 0.016 s
econds
Rows Read: 99, Total Rows Processed: 1599, Total Chunk Time: Less tha
n .001 seconds
```

In [75]:
```
#can remove col quality now
ColsToKeep = setdiff(names(data_classi),c('quality'))
ColsToKeep

data_classi = rxDataStep(inData = data_classi, outFile = 'data_classi_te
mp.xdf',varsToKeep = ColsToKeep, overwrite = TRUE. reportProgress=0)
```

Out[75]:
```
"fixed.acidity"  "volatile.acidity"  "citric.acid"  "residual.sugar"  "chlorides"
"free.sulfur.dioxide"  "total.sulfur.dioxide"  "density"  "pH"  "sulphates"
"alcohol"  "factorQuality"
```

```
Rows Read: 500, Total Rows Processed: 500, Total Chunk Time: 0.009 seco
nds
Rows Read: 500, Total Rows Processed: 1000, Total Chunk Time: Less than
.001 seconds
Rows Read: 500, Total Rows Processed: 1500, Total Chunk Time: 0.016 sec
onds
Rows Read: 99, Total Rows Processed: 1599, Total Chunk Time: Less than
.001 seconds
```

In [46]:
```
dim(data_classi)
#use rxDataStep() to create a col called 'splitcol' to use for splitting
rxDataStep(inData=data_classi,outFile=data_classi,transforms=list(splitc
ol=factor(rbinom(.rxNumRows,1,0.8),labels=c('test','train'))),overwrite=
T)
dim(data_classi)
```

Out[46]: 1599 12

```
Rows Read: 500, Total Rows Processed: 500, Total Chunk Time: 0.012 seco
ndsRows Read: 500, Total Rows Processed: 1000, Total Chunk Time: 0.008
secondsRows Read: 500, Total Rows Processed: 1500, Total Chunk Time: Le
ss than .001 secondsRows Read: 99, Total Rows Processed: 1599, Total Ch
unk Time: 0.015 seconds
```

Out[46]: 1599 13

In [47]:
```
names(data_classi)
#rename the label col 'income' as 'LabelsCol'
names(data_classi)[names(data_classi)=='factorQuality'] = 'LabelsCol'
names(data_classi)
```

Out[47]: "fixed.acidity" "volatile.acidity" "citric.acid" "residual.sugar" "chlorides" "free.sulfur.dioxide" "total.sulfur.dioxide" "density" "pH" "sulphates" "alcohol" "factorQuality" "splitcol"

Out[47]: "fixed.acidity" "volatile.acidity" "citric.acid" "residual.sugar" "chlorides" "free.sulfur.dioxide" "total.sulfur.dioxide" "density" "pH" "sulphates" "alcohol" "LabelsCol" "splitcol"

In [48]:
```
#split using the col "splitcol"
#rxSplit() -- Splits an input '.xdf' file or data frame into multiple '.
xdf' files or a list of data frames.
listofxdfs = rxSplit(data_classi,outFileBase='data_classi_split',outFile
Suffixes=c("Train", "Test"),splitByFactor = "splitcol",overwrite=T )

trainingdata = listofxdfs[[2]]
testdata = listofxdfs[[1]]
dim(trainingdata)
dim(testdata)
dim(data_classi)
```

```
Rows Read: 500, Total Rows Processed: 500, Total Chunk Time: 0.080 seco
ndsRows Read: 500, Total Rows Processed: 1000, Total Chunk Time: 0.082
secondsRows Read: 500, Total Rows Processed: 1500, Total Chunk Time: 0.
088 secondsRows Read: 99, Total Rows Processed: 1599, Total Chunk Time:
0.113 seconds
```

Out[48]: 1272 13

Out[48]: 327 13

Out[48]: 1599 13

In [49]:
```
#collect names of columns (features) to be used for modelling
allfeatures = setdiff(names(data_classi),c('LabelsCol','splitcol'))

#create formula for modelling
formula = as.formula(paste('LabelsCol',paste(allfeatures,collapse=' +
'),sep=' ~ '))
formula
```

Out[49]:
```
LabelsCol ~ fixed.acidity + volatile.acidity + citric.acid +
    residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dio
xide +
    density + pH + sulphates + alcohol
```

In [50]:
```
Algorithms <- c("Decision Forest Classification",
                "Boosted Decision Tree Classification",
                "Decision Tree Classification")
```

Out[50]: "fixed.acidity" "volatile.acidity" "citric.acid" "residual.sugar" "chlorides" "free.sulfur.dioxide" "total.sulfur.dioxide" "density" "pH" "sulphates" "alcohol" "LabelsCol" "splitcol"

In [51]:
```
################################################################################
########
## Decision forest modeling
################################################################################
########
#Decision Forest
#using rxDForest() to build ML model
DForest_model <- rxDForest(formula = formula,
                           data = trainingdata,
                           seed = 10,
                           cp = 0.01,
                           nTree = 50,
                           mTry = 2,
                           overwrite = TRUE,
                           reportProgress = 0)
DForest_model
class(DForest_model) #"rxDForest"
```

Out[51]: "rxDForest"

Out[51]:
```
Call:
rxDForest(formula = formula, data = trainingdata, overwrite = TRUE,
    cp = 0.01, nTree = 50, mTry = 2, seed = 10, reportProgress = 0)


             Type of decision forest: class
                    Number of trees: 50
No. of variables tried at each split: 2

         OOB estimate of error rate: 39.07%
Confusion matrix:
         Predicted
LabelsCol   5   6  7 4 8 3 class.error
        5 395 146  1 0 0 0   0.2712177
        6 145 360  4 0 0 0   0.2927308
        7   5 130 20 0 0 0   0.8709677
        4  22  21  0 0 0 0   1.0000000
        8   0  14  0 0 0 0   1.0000000
        3   7   2  0 0 0 0   1.0000000
```

In [52]:
```
################################################################################
########
## Boosted tree modeling
################################################################################
########
BoostedTree_model = rxBTrees(formula = formula,
                             data = trainingdata,
                             learningRate = 0.2,
                             minSplit = 10,
                             minBucket = 10,
                             nTree = 100,
                             lossFunction = "multinomial",
                             reportProgress = 0)
BoostedTree_model
class(BoostedTree_model)
```

Out[52]:
```
Call:
rxBTrees(formula = formula, data = trainingdata, minSplit = 10,
    minBucket = 10, nTree = 100, lossFunction = "multinomial",
    learningRate = 0.2, reportProgress = 0)


      Loss function of boosted trees: multinomial
       Number of boosting iterations: 100
No. of variables tried at each split: 3

             OOB estimate of deviance: 0.7147638
```

Out[52]: "rxBTrees" "rxDForest"

Out[52]:
```
Call:
rxBTrees(formula = formula, data = trainingdata, minSplit = 10,
    minBucket = 10, nTree = 100, lossFunction = "multinomial",
    learningRate = 0.2, reportProgress = 0)


      Loss function of boosted trees: multinomial
       Number of boosting iterations: 100
No. of variables tried at each split: 3

             OOB estimate of deviance: 0.7147638
```

In [53]:
```
##############################################################################
########
## Decision Tree Modelling
##############################################################################
########

#rxDTree
DTree_model = rxDTree(formula = formula,
                          data = trainingdata,
                          minSplit = 10,
                          minBucket = 10,
                          nTree = 100,
                          reportProgress = 0)
DTree_model
class(DTree_model)
```

Out[53]:
```
Call:
rxDTree(formula = formula, data = trainingdata, minSplit = 10,
    minBucket = 10, reportProgress = 0, nTree = 100)


             Type of decision forest: class
                     Number of trees: 100
No. of variables tried at each split: 0
```

Out[53]: "rxDForest"

```
Error in eval(expr, envir, enclos): object 'DTree_modell' not found
```

Out[53]:
```
Call:
rxBTrees(formula = formula, data = trainingdata, minSplit = 10,
    minBucket = 10, nTree = 100, lossFunction = "multinomial",
    learningRate = 0.2, reportProgress = 0)


      Loss function of boosted trees: multinomial
       Number of boosting iterations: 100
No. of variables tried at each split: 3

            OOB estimate of deviance: 0.7147638
```

In [71]:
```
#Function to compute accuracy of the trained model on the given data
computeaccuracy <- function(ML_model,scoredata){
  if(file.exists("modelout_xdf.xdf") ) {  file.remove("modelout_xdf.xd
f") }
  modelout_xdf = RxXdfData("modelout_xdf.xdf") #initialise xdf object
  rxPredict(ML_model, data = scoredata, outData = modelout_xdf, overwrit
e = TRUE,
            writeModelVars = TRUE, reportProgress = 0)
  #head(modelout_xdf)  #contains the actual and predicted cols

  #get the columns "LabelsCol_Pred" and "LabelsCol" from modelout_xdf
  results_model_df = rxDataStep(inData=modelout_xdf,outFile=NULL,varsToK
eep=c('LabelsCol_Pred','LabelsCol'),reportProgress = 0)
  head(results_model_df)

  actual    = results_model_df$LabelsCol
  predicted = results_model_df$LabelsCol_Pred
  cm = as.matrix(table(Actual=actual, Predicted=predicted)) #create a co
nfusion matrix
  cm
  accuracy = sum(diag(cm)) / sum(cm)
  accuracy
  #cat('The model produced an accuracy = ',accuracy,'\n')
  return(accuracy)
}
# to invoke function:
# computeaccuracy(ML_model,testdata)
# computeaccuracy(ML_model,trainingdata)
# where
# ML_model = DForest_model
# ML_model = BoostedTree_model
# ML_model = DTree_model
```

In [72]:
```
#================================================

ML_model = DForest_model

cat('For Decision Forest: accuracy = ',computeaccuracy(ML_model,training
data),'\n')
cat('For Decision Forest: accuracy on test data = ',computeaccuracy(ML_m
odel,testdata),'\n')
#================================================
ML_model = BoostedTree_model

cat('For Boosted tree: accuracy = ',computeaccuracy(ML_model,trainingdat
a),'\n')
cat('For Boosted tree: accuracy on test data = ',computeaccuracy(ML_mode
l,testdata),'\n')
#================================================
ML_model = DTree_model

cat('For Decision Tree: accuracy = ',computeaccuracy(ML_model,trainingda
ta),'\n')
cat('For Decision Tree: accuracy on test data = ',computeaccuracy(ML_mod
el,testdata),'\n')
```

```
For Decision Forest: accuracy =  0.6320755
For Decision Forest: accuracy on test data =  0.6024465
For Boosted tree: accuracy =  0.697327
For Boosted tree: accuracy on test data =  0.5779817
For Decision Tree: accuracy =  0.75
For Decision Tree: accuracy on test data =  0.5504587
```

In [ ]:
```
##############################################################################
########
## Cleanup
##############################################################################
########
file.remove(temp_train_xdf)
rm(list = ls())
```