Wine Quality Codes

ST443 Machine Learning

ST443 Machine Learning Project - Problem 1

```r
library(ggplot2)
library(corrplot)
library(leaps)
library(glmnet)
library(MASS)
library(gam)
library(tree)
library(randomForest)
library(randomForestExplainer)
library(gbm)
library(xgboost)
library(e1071)
library(scales)
library(ggrepel)
library(neuralnet)
library(ROCR)

flag = "red"
filename <- paste0(
  "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-",
  flag,
  ".csv")
# 6 class
wine <- read.csv(filename, sep=";")

# ggplot(rbind(wine, wine_white),aes(x=quality))+
#   geom_bar(aes(y = (..count..)/sum(..count..)), fill="#FF6666")+
#   scale_x_continuous(breaks = seq(3,9))+
#   ylab("Percent")+
#   xlab("Quality")+
#   theme(panel.background = element_blank(),
#         text = element_text(size=25),
#         axis.text.x = element_text(hjust=1))+
#   geom_text(aes(label = scales::percent(round((..count..)/sum(..count..),4)),
#             y= ((..count..)/sum(..count..))),
#             size=7,
#             stat="count",
#             vjust = -.25)

# # 3 class
# wine$quality <- ifelse(wine$quality<5, 0, ifelse(wine$quality<7, 1, 2))

# 2 class
wine$quality <- ifelse(wine$quality<6, 0L, 1L)

attach(wine)
```

```
# columns
columns.all <- names(wine)
# columns without y
columns <- columns.all[columns.all != "quality"]

# train-test-split
set.seed(2)
train <-sample(1:nrow(wine), round(nrow(wine)*0.9))
wine.test <- wine[-train,]
wine.test.quality <- wine.test$quality

############# have a look at the data #####################

# summary(wine)
# str(wine)
#
# for(name in names(wine)){
#   hist(wine[[name]], main=name)
# }
#
# pairs(wine)

corrplot(cor(wine), method = "shade",
         tl.col = "black")

###########################################################
########################## Tree ###########################
###########################################################

wine2 <- wine
wine2$quality <- as.factor(wine$quality)

################### Simple Tree #####################

trees.wine <- tree(quality ~ ., data = wine2, subset = train)
wine.pred <-predict(trees.wine, wine.test, type="class")
mean(wine.pred!=wine.test.quality)

################### CV Tree ####################

set.seed(2)
cv.wines <-cv.tree(trees.wine)

sde <- sqrt(var(cv.wines$dev)/length(cv.wines$dev))
min_point <- which.min(cv.wines$dev)

plot(cv.wines$size, cv.wines$dev, type="b")
points(which(cv.wines$size==min_point),
       cv.wines$dev[min_point],
       col="red", cex=2, pch=20)

lines(x=1:length(cv.wines$dev),
      y=rep(cv.wines$dev[min_point]+sde,length(cv.wines$dev)),
      type="l", lty=2)


prune.wines <- prune.misclass(trees.wine, best=6)

# plot(prune.wines)
# text(prune.wines, pretty=0)

tree.pred <- predict(prune.wines, wine.test, type="class")
mean(tree.pred!=wine.test.quality)

################### Simple bagging ####################

set.seed(2)
bag.wines <- randomForest(quality~.,
                          data=wine2,
                          subset=train,
                          mtry=11,
                          importance=TRUE)

yhat.bag <- predict(bag.wines, newdata=wine.test)
mean(yhat.bag!=wine.test.quality)
# importance(bag.wines)

################### Random Forest ####################

set.seed(2)
rf.wines <- randomForest(quality~.,
                          data=wine2,
                          subset=train,
                          mtry=6,
                          importance=TRUE,
                          n.tree=1000)
yhat.rf <-predict(rf.wines, newdata=wine.test)
mean(yhat.rf!=wine.test.quality)
varImpPlot(rf.wines)

# if(flag=="red"){
#   color <- "#DC143C"
# }else{
#   color <- "#008080"
# }
#
# mi <- measure_importance(rf.wines)
# ggplot(mi, aes(x=accuracy_decrease, y=gini_decrease)) +
#   geom_point(aes(size=no_of_nodes), color=color) +
#   geom_text_repel(label=mi$variable, size=7, color=color) +
#   theme(text = element_text(size=20),
#         axis.title.x = element_text(colour = color),
#         axis.title.y = element_text(colour = color),
#         axis.text.x = element_text(colour = color),
#         axis.text.y = element_text(colour = color)) +
#   ylab("Gini Decrease") +
#   xlab("Accuracy Decrease") +
#   guides(size=FALSE, color=FALSE)

################### Boosting Tree ####################

set.seed(2)
boost.wine = gbm(quality ~ .,
                 data = wine[train , ],
                 distribution = "bernoulli",
                 n.trees = 1000,
                 interaction.depth = 4)

yhat.boost = predict(boost.wine,
                     newdata = wine.test,
                     n.trees = 1000,
                     type = "response")

mean(as.numeric(yhat.boost>0.5)!=wine.test.quality)

################### XGBoost Tree ####################

xgboost.train <- xgb.DMatrix(data = data.matrix(wine[train, columns]),
                             label = wine[train, ]$quality)

xgboost.valid <- xgb.DMatrix(data = data.matrix(wine[-train, columns]),
                             label = wine[-train, ]$quality)

if(flag=="red"){
  eta <- 0.15
}else{
  eta <- 0.3
}

parameters <- list(
  booster            = "gbtree",
  silent             = 1,
  eta                = eta
)

set.seed(2)
xgb.wine <- xgb.train(parameters, xgboost.train, nrounds = 100)
xgb.pred <- predict(xgb.wine, xgboost.valid)
mean(as.numeric(xgb.pred>0.5)!=wine.test.quality)
xgb.plot.multi.trees(model=xgb.wine, plot_width=500, plot_height=1000)

# draw importance
if(flag=="red"){
  color <- "#DC143C"
}else{
  color <- "#008080"
}
mi_xgb <- xgb.importance(model=xgb.wine)
ggplot(mi_xgb, aes(x=Cover, y=Gain)) +
  geom_point(aes(size=mi_xgb$Frequency), color=color) +
  geom_text_repel(label=mi_xgb$Feature, size=7, color=color) +
  theme(text = element_text(size=20),
        axis.title.x = element_text(colour = color),
        axis.title.y = element_text(colour = color),
        axis.text.x = element_text(colour = color),
        axis.text.y = element_text(colour = color)) +
  ylab("Gain") +
  xlab("Cover") +
  guides(size=FALSE, color=FALSE)

# calculate f1
table <- table(wine.test.quality, as.numeric(xgb.pred>0.5))
recall<-table[2,2]/(table[2,2]+table[2,1])
precison<-table[2,2]/(table[2,2]+table[1,2])
f1 <- 2*precison*recall/(precison+recall)

# draw roc
xgb.prediction <- prediction(xgb.pred, wine.test.quality)
xgb.perf <- performance(xgb.prediction, "tpr", "fpr")
plot(xgb.perf,
     avg="threshold",
     colorize=TRUE,
     lwd=1,
     main="ROC Curve",
     print.cutoffs.at=seq(0, 1, by=0.05),
     text.adj=c(-0.5, 0.5),
     text.cex=0.5,
     cex.lab=1.5,
     cex.axis=1.5)
grid(col="lightgray")
axis(1, at=seq(0, 1, by=0.1))
axis(2, at=seq(0, 1, by=0.1))
abline(v=c(0.1, 0.3, 0.5, 0.7, 0.9), col="lightgray", lty="dotted")
abline(h=c(0.1, 0.3, 0.5, 0.7, 0.9), col="lightgray", lty="dotted")
lines(x=c(0, 1), y=c(0, 1), col="black", lty="dotted")
aucscore <- format(round(slot(performance(xgb.prediction,"auc"),
                              "y.values")[[1]], 2),
                              nsmall = 2)
text(x = 0.4, y=0.6,
     paste("AUC:", aucscore),
     cex=1.5)

############################################################
############# Polynomial-Best Selection-LDA ###############
############################################################

wine1 <- wine

############# add polynomial items #####################

for(name in columns){
  wine1[, paste(name, "2", sep="")] <- wine[, name]**2
  wine1[, paste(name, "3", sep="")] <- wine[, name]**3
}

names_product <- names(sort(importance(rf.wines)[,2], decreasing = TRUE))[1:2]

wine1[, "product"] <- wine1[, names_product[1]] * wine1[, names_product[2]]

wine1.test <- wine1[-train,]

############# lasso selection #####################

x <- model.matrix(quality~.-1, data=wine1)
y <- wine1$quality

set.seed(3)
cv.lasso <-cv.glmnet(x, y, lambda=exp(seq(-3, -12, length.out = 100)), nfolds=10)
plot(cv.lasso)

############# Extract best models #####################

wine_select <- wine1[,dimnames(coef(cv.lasso))[[1]][as.vector(coef(cv.lasso)!=0)][-1]]

############# Logistic Rregresson #####################

glm_fit = glm(quality ~ .,
              data = wine_select,
              subset = train,
              family = binomial)

glm_probs = predict(glm_fit, wine1.test, type = "response")
mean(as.numeric(glm_probs>0.5)==wine.test.quality)


############# LDA #####################

lda_fit = lda(quality ~ .,
              data = wine_select,
              subset = train)

lda_pred = predict(lda_fit, wine1.test)$class
mean(lda_pred == wine.test.quality)

############# QDA #####################

qda_fit = qda(quality ~ .,
              data = wine_select,
              subset = train)

wine_test = wine$quality[-train]
qda_pred = predict(qda_fit, wine1.test)$class
mean(qda_pred == wine.test.quality)

###########################################################
```

```
########################## Spline #########################
###########################################################

############# Natural Spline #####################

formula_ns <- as.formula(paste0("quality~",paste0("ns(",columns,")",collapse="+")))

ns_fit = gam(formula_ns,
             data = wine,
             subset = train,
             family = binomial)

ns_pred = predict(ns_fit, wine.test, type = "response")
mean(as.numeric(ns_pred>0.5)==wine.test.quality)

############# Smoothing Spline #####################

formula_s <- as.formula(paste0("quality~",paste0("s(",columns,")",collapse="+")))

s_fit = gam(formula_s,
              data = wine,
              subset = train,
              family = binomial)

s_pred = predict(s_fit, wine.test, type = "response")
mean(as.numeric(s_pred>0.5)==wine.test.quality)

###########################################################
########################## SVM ############################
###########################################################

svmfit.wine = svm(quality ~ ., data = wine, subset = train)
yhat.svm <- predict(svmfit.wine, newdata = wine.test, type="response")

mean(as.numeric(yhat.svm>0.5)!=wine.test.quality)


wine3 <- wine
wine3$quality <- as.factor(wine3$quality)


# set.seed(2)
# tune.out = tune(svm, quality~., data = wine3[train,], kernel = "radial",
#                 ranges = list(cost = c(5, 5.5, 6, 6.5, 7, 7.5),
#                 gamma = c(0.3, 0.5, 0.8, 1, 1.2)))
# bestmod = tune.out$best.model


if(flag=="red"){
  cost <- 5
  gamma <- 0.4
}else{
  cost <-5.5
  gamma <- 0.5
}

bestmod <- svm(quality~., data = wine3[train,], kernel = "radial",
               cost = cost, gamma = gamma)

ypred = predict(bestmod, wine.test)
table(predict = ypred, truth = wine.test.quality)
mean(ypred!=wine.test.quality)

###########################################################
########################## NNet  ##########################
###########################################################

f <- as.formula(paste("quality ~", paste(columns, collapse = " + ")))

set.seed(2)
network <- neuralnet(f, data = wine[train,], hidden = c(5,3), threshold=0.1,
                     linear.output=FALSE, stepmax=1e7)

net.predict <- compute(network, wine.test[,-12])
net.pred <- sapply(net.predict$net.result,round,digits=0)

mean(net.pred!=wine.test.quality)
plot(network)

###########################################################
########################## Compare  #######################
###########################################################

############ordinary Logistic regression##########
emp_fit = glm(quality ~ .,
              data = wine,
              family = binomial)

summary(emp_fit)
```