Data Mining HW3

# 1. Random Forest Classifier

The response is a categorical variable with 10 classes coded from 0 to 9. All predictors are quantitative.

Construct a random forest classifier, report the test classification error and make the confusion matrix. Note “ranger” is faster than “randomForest”.

### load dataset  
  
student = 20152410  
  
mnist\_train = read.csv('./MNIST\_train\_small.csv', header=TRUE)  
mnist\_test = read.csv('./MNIST\_test\_small.csv', header=TRUE)  
  
train\_data = mnist\_train[, 2:785]  
train\_label = as.factor(mnist\_train$y)  
  
test\_data = mnist\_test[, 2:785]  
test\_label = as.factor(mnist\_test$y)

### random forest classifier  
  
library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Loading required package: ggplot2

library(ranger)

## Warning: package 'ranger' was built under R version 3.6.3

set.seed(student)  
ranger\_model = ranger(x = train\_data,  
 y = train\_label)  
ranger\_model

## Ranger result  
##   
## Call:  
## ranger(x = train\_data, y = train\_label)   
##   
## Type: Classification   
## Number of trees: 500   
## Sample size: 6000   
## Number of independent variables: 784   
## Mtry: 28   
## Target node size: 1   
## Variable importance mode: none   
## Splitrule: gini   
## OOB prediction error: 5.75 %

ranger\_pred = predict(ranger\_model, data=test\_data,  
 num.trees=ranger\_model$num.trees)  
  
ranger\_clf\_error = mean(ranger\_pred$predictions != test\_label)  
cat('Test error of ranger classifier : ', 100\*ranger\_clf\_error, '%')

## Test error of ranger classifier : 6.7 %

ranger\_table = table(ranger\_pred$predictions, test\_label)  
ranger\_cfm = confusionMatrix(ranger\_table, mode='everything')  
ranger\_cfm

## Confusion Matrix and Statistics  
##   
## test\_label  
## 0 1 2 3 4 5 6 7 8 9  
## 0 93 0 3 0 0 2 1 0 0 1  
## 1 0 106 0 0 0 1 0 0 0 3  
## 2 0 0 104 3 0 0 0 2 0 0  
## 3 0 0 2 74 0 2 0 0 2 3  
## 4 0 0 0 0 99 0 1 1 1 1  
## 5 0 0 0 4 0 84 1 0 0 0  
## 6 0 0 0 0 1 3 96 0 1 0  
## 7 0 0 1 2 0 1 0 100 2 1  
## 8 1 0 1 3 0 1 0 1 83 0  
## 9 0 0 0 0 8 1 0 4 1 94  
##   
## Overall Statistics  
##   
## Accuracy : 0.933   
## 95% CI : (0.9157, 0.9477)  
## No Information Rate : 0.111   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9255   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.9894 1.0000 0.9369 0.8605 0.9167 0.8842  
## Specificity 0.9923 0.9955 0.9944 0.9902 0.9955 0.9945  
## Pos Pred Value 0.9300 0.9636 0.9541 0.8916 0.9612 0.9438  
## Neg Pred Value 0.9989 1.0000 0.9921 0.9869 0.9900 0.9879  
## Precision 0.9300 0.9636 0.9541 0.8916 0.9612 0.9438  
## Recall 0.9894 1.0000 0.9369 0.8605 0.9167 0.8842  
## F1 0.9588 0.9815 0.9455 0.8757 0.9384 0.9130  
## Prevalence 0.0940 0.1060 0.1110 0.0860 0.1080 0.0950  
## Detection Rate 0.0930 0.1060 0.1040 0.0740 0.0990 0.0840  
## Detection Prevalence 0.1000 0.1100 0.1090 0.0830 0.1030 0.0890  
## Balanced Accuracy 0.9908 0.9978 0.9657 0.9253 0.9561 0.9393  
## Class: 6 Class: 7 Class: 8 Class: 9  
## Sensitivity 0.9697 0.9259 0.9222 0.9126  
## Specificity 0.9945 0.9922 0.9923 0.9844  
## Pos Pred Value 0.9505 0.9346 0.9222 0.8704  
## Neg Pred Value 0.9967 0.9910 0.9923 0.9899  
## Precision 0.9505 0.9346 0.9222 0.8704  
## Recall 0.9697 0.9259 0.9222 0.9126  
## F1 0.9600 0.9302 0.9222 0.8910  
## Prevalence 0.0990 0.1080 0.0900 0.1030  
## Detection Rate 0.0960 0.1000 0.0830 0.0940  
## Detection Prevalence 0.1010 0.1070 0.0900 0.1080  
## Balanced Accuracy 0.9821 0.9590 0.9573 0.9485

# 2. Boosting Classifier

Construct a boosting classifier, report the test classification error and make the confusion matrix. Note “xgboost” is faster than “gbm”.

### boosting classifier  
  
library(xgboost)

## Warning: package 'xgboost' was built under R version 3.6.3

train\_dmatrix = xgb.DMatrix(data=as.matrix(train\_data), label=as.integer(train\_label)-1)  
test\_dmatrix = xgb.DMatrix(data=as.matrix(test\_data), label=as.integer(test\_label)-1)  
  
xgb\_params = list(eta=0.2,  
 num\_class=length(levels(train\_label)),  
 objective='multi:softmax',  
 eval\_metric='mlogloss')  
  
set.seed(student)  
xgb\_model = xgb.train(data=train\_dmatrix,  
 params=xgb\_params,  
 nrounds=500,  
 early\_stopping\_rounds=20,  
 watchlist=list(val1=train\_dmatrix, val2=test\_dmatrix),  
 verbose=0)  
xgb\_model

## ##### xgb.Booster  
## raw: 1.5 Mb   
## call:  
## xgb.train(params = xgb\_params, data = train\_dmatrix, nrounds = 500,   
## watchlist = list(val1 = train\_dmatrix, val2 = test\_dmatrix),   
## verbose = 0, early\_stopping\_rounds = 20)  
## params (as set within xgb.train):  
## eta = "0.2", num\_class = "10", objective = "multi:softmax", eval\_metric = "mlogloss", silent = "1"  
## xgb.attributes:  
## best\_iteration, best\_msg, best\_ntreelimit, best\_score, niter  
## callbacks:  
## cb.evaluation.log()  
## cb.early.stop(stopping\_rounds = early\_stopping\_rounds, maximize = maximize,   
## verbose = verbose)  
## # of features: 784   
## niter: 130  
## best\_iteration : 110   
## best\_ntreelimit : 110   
## best\_score : 0.216168   
## nfeatures : 784   
## evaluation\_log:  
## iter val1\_mlogloss val2\_mlogloss  
## 1 1.617869 1.703273  
## 2 1.285146 1.410190  
## ---   
## 129 0.002439 0.216710  
## 130 0.002420 0.216638

xgb\_pred = as.factor(predict(xgb\_model, newdata=test\_dmatrix))  
  
xgb\_clf\_error = mean(xgb\_pred != test\_label)  
cat('Test error of xgboost classifier : ', 100\*xgb\_clf\_error, '%')

## Test error of xgboost classifier : 6.5 %

xgb\_table = table(xgb\_pred, test\_label)  
xgb\_cfm = confusionMatrix(xgb\_table, mode='everything')  
xgb\_cfm

## Confusion Matrix and Statistics  
##   
## test\_label  
## xgb\_pred 0 1 2 3 4 5 6 7 8 9  
## 0 93 0 2 0 0 3 1 1 0 1  
## 1 0 106 0 0 0 0 0 0 0 2  
## 2 0 0 105 3 0 1 0 1 3 0  
## 3 0 0 2 76 0 1 1 0 1 1  
## 4 0 0 0 1 101 0 1 0 1 3  
## 5 0 0 0 1 0 83 1 0 0 2  
## 6 0 0 0 1 0 2 94 0 0 0  
## 7 0 0 0 0 0 1 0 100 2 0  
## 8 1 0 2 3 0 3 1 1 83 0  
## 9 0 0 0 1 7 1 0 5 0 94  
##   
## Overall Statistics  
##   
## Accuracy : 0.935   
## 95% CI : (0.9179, 0.9495)  
## No Information Rate : 0.111   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9277   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.9894 1.0000 0.9459 0.8837 0.9352 0.8737  
## Specificity 0.9912 0.9978 0.9910 0.9934 0.9933 0.9956  
## Pos Pred Value 0.9208 0.9815 0.9292 0.9268 0.9439 0.9540  
## Neg Pred Value 0.9989 1.0000 0.9932 0.9891 0.9922 0.9869  
## Precision 0.9208 0.9815 0.9292 0.9268 0.9439 0.9540  
## Recall 0.9894 1.0000 0.9459 0.8837 0.9352 0.8737  
## F1 0.9538 0.9907 0.9375 0.9048 0.9395 0.9121  
## Prevalence 0.0940 0.1060 0.1110 0.0860 0.1080 0.0950  
## Detection Rate 0.0930 0.1060 0.1050 0.0760 0.1010 0.0830  
## Detection Prevalence 0.1010 0.1080 0.1130 0.0820 0.1070 0.0870  
## Balanced Accuracy 0.9903 0.9989 0.9685 0.9386 0.9642 0.9346  
## Class: 6 Class: 7 Class: 8 Class: 9  
## Sensitivity 0.9495 0.9259 0.9222 0.9126  
## Specificity 0.9967 0.9966 0.9879 0.9844  
## Pos Pred Value 0.9691 0.9709 0.8830 0.8704  
## Neg Pred Value 0.9945 0.9911 0.9923 0.9899  
## Precision 0.9691 0.9709 0.8830 0.8704  
## Recall 0.9495 0.9259 0.9222 0.9126  
## F1 0.9592 0.9479 0.9022 0.8910  
## Prevalence 0.0990 0.1080 0.0900 0.1030  
## Detection Rate 0.0940 0.1000 0.0830 0.0940  
## Detection Prevalence 0.0970 0.1030 0.0940 0.1080  
## Balanced Accuracy 0.9731 0.9613 0.9551 0.9485