GermanCredit\_long\_bagged\_trees.R

#setwd("")  
  
########################################################################  
#DEfinition bagged tree model  
########################################################################  
library(ipred)  
#number of bagged trees can be specified using the nbagg parameter  
#here we will use the default (25)  
  
#If we want to estimate the model's accuracy using the "out-of-bag" (OOB) samples  
#we can set the the coob parameter to TRUE  
  
#The OOB samples are the training obsevations that were not selected into the bootstrapped sample (used in training)  
#Since these observations were not used in training, we can use them instead to evaluate the accuracy of the model  
#done automatically inside the bagging() function  
  
credit <- read.csv("credit.csv", stringsAsFactors = TRUE)  
########################################################################  
#Split data in 80% 20%  
########################################################################  
# Total number of rows in the credit data frame  
n <- nrow(credit)  
# Number of rows for the training set (80% of the dataset)  
n\_train <- round(0.8 \* n)   
# Create a vector of indices which is an 80% random sample  
set.seed(123)  
train\_indices <- sample(1:n, n\_train)  
# Subset the credit data frame to training indices only  
credit\_train <- credit[train\_indices, ]   
# Exclude the training indices to create the test set  
credit\_test <- credit[-train\_indices, ]  
  
########################################################################  
#Train a bagged tree model  
########################################################################  
  
# Bagging is a randomized model, so let's set a seed (123) for reproducibility  
set.seed(123)  
  
# Train a bagged model  
credit\_model <- bagging(formula = default ~ .,   
 data = credit\_train,  
 coob = TRUE)  
  
# Print the model  
print(credit\_model)

##   
## Bagging classification trees with 25 bootstrap replications   
##   
## Call: bagging.data.frame(formula = default ~ ., data = credit\_train,   
## coob = TRUE)  
##   
## Out-of-bag estimate of misclassification error: 0.2788

########################################################################  
#Prediction and confusion matrix  
########################################################################  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

# Generate predicted classes using the model object  
class\_prediction <- predict(object = credit\_model,   
 newdata = credit\_test,   
 type = "class") # return classification labels  
  
# Print the predicted classes  
print(class\_prediction)

## [1] no yes yes no no yes no yes no no no yes no yes no no no   
## [18] no no no no no no yes no no no yes no yes yes yes no no   
## [35] no no no no no no no no no yes no no no yes no yes yes  
## [52] no no yes no no no no no no no no no no no yes no no   
## [69] no no yes no no yes no no no no no no no no no no no   
## [86] no no no no no yes no yes no no no no yes no no no no   
## [103] no no yes no no no no no no no no no no no no no no   
## [120] no no no yes no no no no no no no no no no no no no   
## [137] no no no no yes no yes no yes no no no no no no no yes  
## [154] no no no no no no no no yes no no no no yes yes no no   
## [171] no no yes yes no no no no no no no yes no no no no yes  
## [188] no no no no yes no no no no yes no no yes  
## Levels: no yes

# Calculate the confusion matrix for the test set  
confusionMatrix(data = class\_prediction,   
 reference = credit\_test$default)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 126 36  
## yes 12 26  
##   
## Accuracy : 0.76   
## 95% CI : (0.6947, 0.8174)  
## No Information Rate : 0.69   
## P-Value [Acc > NIR] : 0.0178277   
##   
## Kappa : 0.3721   
## Mcnemar's Test P-Value : 0.0009009   
##   
## Sensitivity : 0.9130   
## Specificity : 0.4194   
## Pos Pred Value : 0.7778   
## Neg Pred Value : 0.6842   
## Prevalence : 0.6900   
## Detection Rate : 0.6300   
## Detection Prevalence : 0.8100   
## Balanced Accuracy : 0.6662   
##   
## 'Positive' Class : no   
##

#Accuracy 0.76  
  
########################################################################  
#Predict on a test set and compute AUC  
########################################################################  
#In binary classification problems, we can predict numeric values instead of class labels  
#In fact, class labels are created only after you use the model to predict a raw, numeric, predicted value for a test point.  
  
#The predicted label is generated by applying a threshold to the predicted value,   
#all tests points with predicted value greater than that threshold get a predicted label of "1"  
#AUC is a common metric for evaluating the discriminatory ability of a binary classification model.  
  
library(Metrics)

##   
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':  
##   
## precision, recall

# Generate predictions on the test set  
pred <- predict(object = credit\_model,  
 newdata = credit\_test,  
 type = "prob")  
# pred is a matrix  
class(pred)

## [1] "matrix"

# Look at the pred format  
head(pred)

## no yes  
## [1,] 0.96 0.04  
## [2,] 0.28 0.72  
## [3,] 0.36 0.64  
## [4,] 0.76 0.24  
## [5,] 0.92 0.08  
## [6,] 0.48 0.52

# Compute the AUC (`actual` must be a binary (or 1/0 numeric) vector)  
auc(actual = ifelse(credit\_test$default == "yes", 1, 0),   
 predicted = pred[,"yes"])

## [1] 0.7809724

########################################################################  
#Cross-validate a bagged tree model in caret  
########################################################################  
# Specify the training configuration  
ctrl <- trainControl(method = "cv", # Cross-validation  
 number = 5, # 5 folds  
 classProbs = TRUE, # For AUC  
 summaryFunction = twoClassSummary) # For AUC  
  
# Cross validate the credit model using "treebag" method;   
# Track AUC (Area under the ROC curve)  
set.seed(1) # for reproducibility  
credit\_caret\_model <- train(default ~ .,  
 data = credit\_train,   
 method = "treebag",  
 metric = "ROC",  
 trControl = ctrl)  
  
# Look at the model object  
print(credit\_caret\_model)

## Bagged CART   
##   
## 800 samples  
## 16 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 641, 640, 640, 639, 640   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.7203687 0.8275126 0.4417553

# Inspect the contents of the model list   
names(credit\_caret\_model)

## [1] "method" "modelInfo" "modelType" "results"   
## [5] "pred" "bestTune" "call" "dots"   
## [9] "metric" "control" "finalModel" "preProcess"   
## [13] "trainingData" "resample" "resampledCM" "perfNames"   
## [17] "maximize" "yLimits" "times" "levels"   
## [21] "terms" "coefnames" "contrasts" "xlevels"

# Print the CV AUC  
credit\_caret\_model$results[,"ROC"]

## [1] 0.7203687

#AUC is 0.72  
  
########################################################################  
#Generate predictions from the caret model  
########################################################################  
  
# Generate predictions on the test set  
pred <- predict(object = credit\_caret\_model,   
 newdata = credit\_test,  
 type = "prob")  
  
#saveRDS(pred[,"yes"], file = "bag\_preds")  
  
# Compute the AUC (`actual` must be a binary (or 1/0 numeric) vector)  
auc(actual = ifelse(credit\_test$default == "yes", 1, 0),   
 predicted = pred[,"yes"])

## [1] 0.7762389

#AUC is 0.77  
  
########################################################################  
#Compare test set performance to CV performance  
########################################################################  
#Lastly, we will print the 5-fold cross-validated estimate of AUC that is stored within the credit\_caret\_model  
#This number will be a more accurate estimate of the true model performance since we have averaged the performance over five models instead of just one.  
#When using small data, it's recommended to use cross-validated estimates of performance because they are more stable  
  
#object stores the test set AUC from the model trained using the ipred::bagging() function.  
credit\_ipred\_model\_test\_auc <- 0.7762389  
  
#object stores the test set AUC from the model trained using the caret::train() function with method = "treebag"  
credit\_caret\_model\_test\_auc <- 0.76  
  
# Print ipred::bagging test set AUC estimate  
print(credit\_ipred\_model\_test\_auc)

## [1] 0.7762389

# Print caret "treebag" test set AUC estimate  
print(credit\_caret\_model\_test\_auc)

## [1] 0.76

# Compare to caret 5-fold cross-validated AUC  
credit\_caret\_model$results[, "ROC"]

## [1] 0.7203687