manual\_crossValidation.R

Norbert

Tue Feb 20 23:46:50 2018

library(LiblineaR)

## Warning: package 'LiblineaR' was built under R version 3.3.3

library(readr)

## Warning: package 'readr' was built under R version 3.3.3

library('e1071')

## Warning: package 'e1071' was built under R version 3.3.3

library(MASS)

## Warning: package 'MASS' was built under R version 3.3.3

library(caret)

## Warning: package 'caret' was built under R version 3.3.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.3

library(adabag)

## Warning: package 'adabag' was built under R version 3.3.3

## Loading required package: rpart

## Warning: package 'rpart' was built under R version 3.3.3

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.3.3

## Loading required package: doParallel

## Warning: package 'doParallel' was built under R version 3.3.3

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 3.3.3

## Loading required package: parallel

#set seed, create sample   
set.seed(2017)  
samp <- sample(1:155113, 100000)  
  
# import data  
labels <- read.csv("C:/Users/Norbert/Desktop/TWE output/new\_labels.csv", header = TRUE)[samp,]  
data <- read.csv("C:/Users/Norbert/Desktop/TWE output/minMaxDoc\_dim400.csv", header = TRUE)[samp,]  
  
  
# LDA Method = moment Manual 10-fold Cross validation  
# Set random seed and optionally shuffle rows  
set.seed(2017)  
  
# Step 2: Create 10 equally size folds  
folds <- cut(seq(1,nrow(data)),breaks=10,labels=FALSE)  
  
# Step 3: set up array to store error rate results  
errorRate <- NA  
pred\_recall <- NA  
pred\_precision <- NA  
cvFold\_predictions <- NA  
  
# Step 4: Perform 10 fold cross validation  
print("First Level Classifier - LDA: 10 fold cross validation")

## [1] "First Level Classifier - LDA: 10 fold cross validation"

for(i in 1:10){  
 # Segment your data by fold using the which() function   
 testIndexes <- which(folds==i,arr.ind=TRUE)  
   
 # inner train sets  
 cvData\_train <- data[-testIndexes,]  
 cvLabels\_train <- labels[-testIndexes]  
   
 # inner test sets  
 cvData\_test <- data[testIndexes,]  
 cvLabels\_test <- labels[testIndexes]  
   
 # fit lda model  
 ldaModel <- lda(formula = factor(cvLabels\_train)~., data = cvData\_train)   
   
 # predicted on the whole data set  
 pred\_lda <- predict(ldaModel, newdata = cvData\_test)  
 cvFold\_predictions[(((i-1)\*10000) + 1):(i\*10000)] <- pred\_lda$class  
   
 # calculate error rate  
 errorRate[i] <- (nrow(cvData\_test)-sum(diag(table(pred\_lda$class, cvLabels\_test))))/nrow(cvData\_test)  
 confMatr <- table(pred\_lda$class, cvLabels\_test)  
   
 # Precision for predator = TN / (TN+FN)  
 # Recall for predator = TN / (TN+FP)  
 pred\_precision[i] <- confMatr[2,2] / (confMatr[2,2] + confMatr[1,2])  
 pred\_recall[i] <- confMatr[2,2] / (confMatr[2,2] + confMatr[2,1])  
   
 # print info fir eeac iteration  
 cat("Fold ", i , " Error rate: ", errorRate[i], "\n")  
 cat("Fold ", i , " Precision rate: ", pred\_precision[i], "\n")  
 cat("Fold ", i , " Recall rate: ", pred\_recall[i], "\n")  
}

## Fold 1 Error rate: 0.0045   
## Fold 1 Precision rate: 0.9814815   
## Fold 1 Recall rate: 0.5463918   
## Fold 2 Error rate: 0.0052   
## Fold 2 Precision rate: 0.8518519   
## Fold 2 Recall rate: 0.5111111   
## Fold 3 Error rate: 0.0057   
## Fold 3 Precision rate: 0.8846154   
## Fold 3 Recall rate: 0.4742268   
## Fold 4 Error rate: 0.0037   
## Fold 4 Precision rate: 0.984127   
## Fold 4 Recall rate: 0.6326531   
## Fold 5 Error rate: 0.005   
## Fold 5 Precision rate: 0.9354839   
## Fold 5 Recall rate: 0.5576923   
## Fold 6 Error rate: 0.0056   
## Fold 6 Precision rate: 0.8947368   
## Fold 6 Recall rate: 0.5049505   
## Fold 7 Error rate: 0.0055   
## Fold 7 Precision rate: 0.9148936   
## Fold 7 Recall rate: 0.4574468   
## Fold 8 Error rate: 0.005   
## Fold 8 Precision rate: 0.9473684   
## Fold 8 Recall rate: 0.5346535   
## Fold 9 Error rate: 0.0042   
## Fold 9 Precision rate: 0.8809524   
## Fold 9 Recall rate: 0.5   
## Fold 10 Error rate: 0.0056   
## Fold 10 Precision rate: 0.8548387   
## Fold 10 Recall rate: 0.53

avgErrorRate\_ldaMoment <- mean(errorRate)  
avgPrecision\_ldaMoment <- mean(pred\_precision)  
avgRecall\_ldaMoment <- mean(pred\_recall)  
  
ldaPred\_cv <- factor(cvFold\_predictions - 1) # classes were 1 and 2, so subtract 1 to have the usual 0 and 1 classes  
  
print("Prediction for left out subsets inside k-fold CV for LDA:")

## [1] "Prediction for left out subsets inside k-fold CV for LDA:"

table(ldaPred\_cv, labels)

## labels  
## ldaPred\_cv 0 1  
## 0 98997 47  
## 1 453 503

# indexing the observations labeled as 1 (predator)  
index <- which(ldaPred\_cv == 1)  
predicted1\_lda <- data[index,]  
labeled1\_lda <- labels[index]

boosting\_df <- data.frame(predicted1\_lda, Y = factor(labeled1\_lda))  
print("Second Level Classifier - Version A - Boosting: 10 fold cross validation")

## [1] "Second Level Classifier - Version A - Boosting: 10 fold cross validation"

boosting\_CV <- boosting.cv(formula = Y~., data=boosting\_df, v = 10, boos = TRUE, mfinal = 300)

## i: 1 Wed Feb 21 00:45:25 2018   
## i: 2 Wed Feb 21 00:53:42 2018   
## i: 3 Wed Feb 21 01:01:50 2018   
## i: 4 Wed Feb 21 01:09:59 2018   
## i: 5 Wed Feb 21 01:18:07 2018   
## i: 6 Wed Feb 21 01:26:14 2018   
## i: 7 Wed Feb 21 01:34:22 2018   
## i: 8 Wed Feb 21 01:42:36 2018   
## i: 9 Wed Feb 21 01:50:46 2018   
## i: 10 Wed Feb 21 01:58:53 2018

print("Prediction for left out subsets inside k-fold CV for Boosting:")

## [1] "Prediction for left out subsets inside k-fold CV for Boosting:"

boosting\_CV$confusion

## Observed Class  
## Predicted Class 0 1  
## 0 332 90  
## 1 121 413

#Mapping back (optional)  
finalPred\_withBoosting <- ldaPred\_cv  
  
for(i in 1:length(index)){  
 if(boosting\_CV$class[i] == 0){  
 finalPred\_withBoosting[index[i]] = 0  
 }  
}  
  
print("Second Level Classifier - Version A - Boosting: Mapping back (optional) ")

## [1] "Second Level Classifier - Version A - Boosting: Mapping back (optional) "

confusionMatrix(data = finalPred\_withBoosting, reference = labels)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 99329 137  
## 1 121 413  
##   
## Accuracy : 0.9974   
## 95% CI : (0.9971, 0.9977)  
## No Information Rate : 0.9945   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7607   
## Mcnemar's Test P-Value : 0.3504   
##   
## Sensitivity : 0.9988   
## Specificity : 0.7509   
## Pos Pred Value : 0.9986   
## Neg Pred Value : 0.7734   
## Prevalence : 0.9945   
## Detection Rate : 0.9933   
## Detection Prevalence : 0.9947   
## Balanced Accuracy : 0.8748   
##   
## 'Positive' Class : 0   
##

# Step 1: Set random seed and optionally shuffle rows  
set.seed(2017)  
  
# Step 2: Create 10 equally size folds  
folds\_svm <- cut(seq(1,nrow(predicted1\_lda)),breaks=10,labels=FALSE)  
  
# Step 3: set up array to store MSE results  
errorRate\_svm <- NA  
pred\_recall\_svm <- NA  
pred\_precision\_svm <- NA  
cvFold\_predictions\_svm <- NA  
ind = 1  
  
# Step 4: Perform 10 fold cross validation  
print("Second Level Classifier - Version B - SVM: 10 fold cross validation ")

## [1] "Second Level Classifier - Version B - SVM: 10 fold cross validation "

for(i in 1:10){  
 #Segment your data by fold using the which() function   
 testIndexes <- which(folds\_svm==i,arr.ind=TRUE)  
 cvData\_train <- predicted1\_lda[-testIndexes,]  
 cvLabels\_train <- labeled1\_lda[-testIndexes]  
   
 cvData\_test <- predicted1\_lda[testIndexes,]  
 cvLabels\_test <- labeled1\_lda[testIndexes]  
   
 libLinear <- LiblineaR(data = cvData\_train, target = factor(cvLabels\_train), type = 3)  
 predictLibLinear <- predict(libLinear, as.matrix(cvData\_test))  
   
 # predicted on the whole data set  
 cvFold\_predictions\_svm[ind:(ind-1+length(testIndexes))] <- predictLibLinear$predictions  
 ind <- ind + length(testIndexes)  
   
 # calculate error rate  
 errorRate\_svm[i] <- (nrow(cvData\_test)-sum(diag(table(cvLabels\_test, predictLibLinear$predictions))))/nrow(cvData\_test)  
   
 # Precision for predator = TN / (TN+FN)  
 # Recall for predator = TN / (TN+FP)  
 confMatr <- table(predictLibLinear$predictions, cvLabels\_test)  
 pred\_precision\_svm[i] <- confMatr[2,2] / (confMatr[2,2] + confMatr[1,2])  
 pred\_recall\_svm[i] <- confMatr[2,2] / (confMatr[2,2] + confMatr[2,1])  
   
 # print info fir each iteration  
 cat("Fold ", i , " Error rate: ", errorRate\_svm[i], "\n")  
 cat("Fold ", i , " Precision rate: ", pred\_precision\_svm[i], "\n")  
 cat("Fold ", i , " Recall rate: ", pred\_recall\_svm[i], "\n")  
   
}

## Fold 1 Error rate: 0.28125   
## Fold 1 Precision rate: 0.8461538   
## Fold 1 Recall rate: 0.6984127   
## Fold 2 Error rate: 0.3229167   
## Fold 2 Precision rate: 0.8571429   
## Fold 2 Recall rate: 0.6363636   
## Fold 3 Error rate: 0.3263158   
## Fold 3 Precision rate: 0.8444444   
## Fold 3 Recall rate: 0.6129032   
## Fold 4 Error rate: 0.2604167   
## Fold 4 Precision rate: 0.8064516   
## Fold 4 Recall rate: 0.7936508   
## Fold 5 Error rate: 0.3473684   
## Fold 5 Precision rate: 0.8076923   
## Fold 5 Recall rate: 0.6461538   
## Fold 6 Error rate: 0.2083333   
## Fold 6 Precision rate: 0.9387755   
## Fold 6 Recall rate: 0.7301587   
## Fold 7 Error rate: 0.3157895   
## Fold 7 Precision rate: 0.8409091   
## Fold 7 Recall rate: 0.6166667   
## Fold 8 Error rate: 0.2916667   
## Fold 8 Precision rate: 0.7959184   
## Fold 8 Recall rate: 0.6842105   
## Fold 9 Error rate: 0.2947368   
## Fold 9 Precision rate: 0.8235294   
## Fold 9 Recall rate: 0.6885246   
## Fold 10 Error rate: 0.2395833   
## Fold 10 Precision rate: 0.82   
## Fold 10 Recall rate: 0.7454545

avgErrorRate\_svm <- mean(errorRate\_svm)  
avgPrecision\_svm <- mean(pred\_precision\_svm)  
avgRecall\_svm <- mean(pred\_recall\_svm)  
  
svmPred\_cv <- factor(cvFold\_predictions\_svm - 1) # classes were 1 and 2, so subtract 1 to have the usual 0 and 1 classes  
  
print("Prediction for left out subsets inside k-fold CV for SVM: ")

## [1] "Prediction for left out subsets inside k-fold CV for SVM: "

table(svmPred\_cv, labeled1\_lda)

## labeled1\_lda  
## svmPred\_cv 0 1  
## 0 259 82  
## 1 194 421

#Mapping back (optional)  
finalPred\_withSVM <- ldaPred\_cv  
  
for(i in 1:length(index)){  
 if(svmPred\_cv[i] == 0){  
 finalPred\_withSVM[index[i]] = 0  
 }  
}  
  
print("Second Level Classifier - Version B - SVM: Mapping back (optional) ")

## [1] "Second Level Classifier - Version B - SVM: Mapping back (optional) "

confusionMatrix(data = finalPred\_withSVM, reference = labels)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 99256 129  
## 1 194 421  
##   
## Accuracy : 0.9968   
## 95% CI : (0.9964, 0.9971)  
## No Information Rate : 0.9945   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7211   
## Mcnemar's Test P-Value : 0.0003694   
##   
## Sensitivity : 0.9980   
## Specificity : 0.7655   
## Pos Pred Value : 0.9987   
## Neg Pred Value : 0.6846   
## Prevalence : 0.9945   
## Detection Rate : 0.9926   
## Detection Prevalence : 0.9939   
## Balanced Accuracy : 0.8818   
##   
## 'Positive' Class : 0   
##