Classification and Regression Trees (CART)

Replication-based stagewise additive modeling (RSAM)

Simulation-based settings

Article: Lizbeth Naranjo, Carlos J. Perez, Daniel F. Merino (2025). A data ensemble-based approach for detecting vocal disorders using replicated acoustic biomarkers from electroglottography.
Sensing and Bio-Sensing Research Journal, vol, num, pages.

Data

```
## read data
datos2 <- read.csv(paste0(address,"data_simulated.csv"),
                   sep = ";",header=TRUE, dec=".")

archivo = "RSAM_crossval_strata_CART_simula"   ## name of the files to save results
```

```
dim(datos2)
```

```
[1] 900  24
```

```
summary(datos2)
```

```
       V1                  V2                  V3                  V4
 Min.   :-2.46818   Min.   :-2.8869   Min.   :-2.3041   Min.   :-2.23859
 1st Qu.:-0.64472   1st Qu.:-0.4958   1st Qu.:-0.1822   1st Qu.: 0.05472
 Median : 0.02661   Median : 0.2384   Median : 0.6347   Median : 1.09540
 Mean   :-0.01209   Mean   : 0.2109   Mean   : 0.6358   Mean   : 1.05924
 3rd Qu.: 0.60494   3rd Qu.: 0.9120   3rd Qu.: 1.3916   3rd Qu.: 1.98603
 Max.   : 2.62886   Max.   : 2.9954   Max.   : 4.0518   Max.   : 4.75694
       V5                  V6                  V7                  V8
 Min.   :-2.9530   Min.   :-2.1113   Min.   :-3.0853   Min.   :-2.236
 1st Qu.: 0.2255   1st Qu.: 0.6053   1st Qu.: 0.9834   1st Qu.: 1.317
 Median : 1.2606   Median : 1.8159   Median : 2.4098   Median : 2.774
 Mean   : 1.4013   Mean   : 1.9805   Mean   : 2.5094   Mean   : 2.689
 3rd Qu.: 2.4265   3rd Qu.: 3.3161   3rd Qu.: 3.9536   3rd Qu.: 3.962
 Max.   : 7.2912   Max.   : 7.2158   Max.   : 8.0646   Max.   : 7.362
       V9                 V10                 V11                 V12
 Min.   :-2.530   Min.   :-1.322   Min.   :-1.068   Min.   :-0.9634
 1st Qu.: 1.402   1st Qu.: 1.986   1st Qu.: 2.123   1st Qu.: 1.8628
 Median : 2.667   Median : 3.055   Median : 3.296   Median : 2.9238
 Mean   : 2.573   Mean   : 3.064   Mean   : 3.388   Mean   : 2.9824
```

```
 3rd Qu.: 3.817   3rd Qu.: 4.127   3rd Qu.: 4.546   3rd Qu.: 4.0124
 Max.   : 7.988   Max.   : 7.115   Max.   : 7.231   Max.   : 7.0910
      V13              V14              V15              V16
 Min.   :-1.885   Min.   :-2.079   Min.   :-2.269   Min.   :-2.1259
 1st Qu.: 1.655   1st Qu.: 1.313   1st Qu.: 1.112   1st Qu.: 0.5982
 Median : 3.038   Median : 2.747   Median : 2.537   Median : 1.9031
 Mean   : 2.805   Mean   : 2.623   Mean   : 2.648   Mean   : 1.9893
 3rd Qu.: 4.087   3rd Qu.: 3.928   3rd Qu.: 4.098   3rd Qu.: 3.3091
 Max.   : 6.597   Max.   : 6.704   Max.   : 8.223   Max.   : 7.2479
      V17               V18               V19               V20
 Min.   :-3.18364   Min.   :-2.7850   Min.   :-3.8817   Min.   :-2.7783
 1st Qu.:-0.01008   1st Qu.:-0.1074   1st Qu.:-0.1600   1st Qu.:-0.2862
 Median : 1.08517   Median : 0.8566   Median : 0.6810   Median : 0.4368
 Mean   : 1.25594   Mean   : 0.9245   Mean   : 0.6666   Mean   : 0.3959
 3rd Qu.: 2.51669   3rd Qu.: 1.9699   3rd Qu.: 1.5111   3rd Qu.: 1.1143
 Max.   : 6.44674   Max.   : 5.6873   Max.   : 4.1971   Max.   : 4.0928
      V21                 ID              rep        status
 Min.   :-2.43443   Min.   :  1.00   Min.   :1   Min.   :1
 1st Qu.:-0.67873   1st Qu.: 75.75   1st Qu.:1   1st Qu.:1
 Median : 0.06097   Median :150.50   Median :2   Median :2
 Mean   : 0.01474   Mean   :150.50   Mean   :2   Mean   :2
 3rd Qu.: 0.67792   3rd Qu.:225.25   3rd Qu.:3   3rd Qu.:3
 Max.   : 3.26405   Max.   :300.00   Max.   :3   Max.   :3
```

```
head(datos2)
```

```
          V1        V2         V3         V4         V5         V6          V7
1  1.1541136  1.206173 -0.9686025 0.02312405  2.3165891 -0.3054029 -1.61019789
2  1.1541136  1.485269 -0.4104103 0.86041236  3.4329735  1.0900776  0.06437873
3  1.1541136  1.485269 -0.4104103 0.86041236  3.4329735  0.3691737 -1.37742906
4 -0.6443284 -1.553137 -1.5977095 1.80509752 -0.4816474  1.3936230  2.15860994
5 -0.6443284 -1.326381 -1.1441960 2.48536784  0.4253797  2.5274069  3.51915059
6 -0.6443284 -1.326381 -1.1441960 2.48536784  0.4253797  1.7541637  1.97266414
         V8        V9      V10      V11      V12      V13      V14      V15
1 2.0685608 2.6366958 5.544770 4.434717 4.987851 3.720234 3.668854 3.077667
2 3.4640413 3.7530802 6.102962 4.434717 4.429659 2.603849 2.273374 1.403090
3 1.3013296 0.8694646 3.219346 1.551101 2.987851 2.603849 3.715182 4.286706
4 2.1574187 3.9048461 6.289806 7.142163 6.178932 4.254271 3.944702 2.582941
5 3.2912026 4.8118732 6.743320 7.142163 5.725419 3.347244 2.810918 1.222400
6 0.9714729 1.7189003 3.650347 4.049190 4.178932 3.347244 4.357404 4.315373
         V16       V17         V18      V19        V20        V21 ID rep status
1 -0.3964881 1.5892168  0.48317124 1.863786 -0.8284375 -0.1784143  1   1      1
2 -1.7919686 0.4728324 -0.35411707 1.305594 -1.1075336 -0.1784143  2   1      2
3  1.0916470 3.3564480  1.80859462 2.747402 -0.3866297 -0.1784143  3   1      3
4  0.2449769 2.6747610  0.70607137 1.582024 -2.1536013 -1.0602656  1   2      1
5 -0.8888070 1.7677339  0.02580105 1.128511 -2.3803581 -1.0602656  2   2      2
6  2.2041659 4.8607067  2.34553072 2.674997 -1.6071148 -1.0602656  3   2      3
```

```
datos2 <- as.data.frame(datos2)
datos2$ID_fact = as.factor(datos2$ID)   ## categorical ID of the subject
datos2$STATUS_fact = as.factor(datos2$status)   ## categorical response variable
table(datos2$STATUS_fact)
```

```
  1   2   3
```

```
300 300 300
```

```
## data set
trainc <- datos2 %>% select(-status,-ID)
```

Crossvalidation

Training and testing data subsets

```
## Select data: 75% training & 25% testing stratified per category
SIM = 100   ## repeat N times the cross-validation process
N = 300   ## sample size
Nfit = 225   ## sample size for training subset
Ntest = 75   ## sample size for testing subset
Ncat = 100   ## sample size per category
Ncatfit = 75   ## training per category
Ncattest = 25   ## testing per category
FIT <- matrix(0,SIM,Nfit)   ## training subsets
TEST <- matrix(0,SIM,Ntest)   ## testing subsets

categoria = trainc %>% filter(rep==1) %>% select(STATUS_fact)
categoria = as.numeric(categoria$STATUS_fact)
id = 1:N
set.seed(12345)
for(si in 1:SIM){
  for(j in 1:3){
    idcat = id[categoria==j]   ## stratified per category j
    ran0 = sample(idcat, size=Ncatfit, replace=FALSE)

    FIT[si,(j-1)*Ncatfit+1:Ncatfit] <- sort(ran0)
    TEST[si,(j-1)*Ncattest+1:Ncattest] <- setdiff(idcat,ran0)
} }
```

Classification metrics for models predicting nominal outcomes

```
## Functions to compute classification metrics
## Ytrue = true response variable
## Ypred = predicted outcome
## cat = category
## TP = true positive
## TN = true negative
## FP = false positive
## FN = false negative

## Function to compute the precision per class=cat
fn_precision_class <- function(Ytrue,Ypred,cat){
  TP = sum(Ypred[Ytrue==cat]==cat)
  FP = sum(Ypred[Ytrue!=cat]==cat)
  precision = TP/(TP+FP)
  return(precision)
}

## Function to compute the recall per class=cat
fn_recall_class <- function(Ytrue,Ypred,cat){ ## cat==category
  TP = sum(Ypred[Ytrue==cat]==cat)
  FN = sum(Ypred[Ytrue==cat]!=cat)
  recall = TP/(TP+FN)
  return(recall)
}

## Function to compute the F1-score per class=cat
fn_f1score_class <- function(Ytrue,Ypred,cat){ ## cat==category
  TP = sum(Ypred[Ytrue==cat]==cat)
  FP = sum(Ypred[Ytrue!=cat]==cat)
  FN = sum(Ypred[Ytrue==cat]!=cat)
  precision = TP/(TP+FP)
  recall = TP/(TP+FN)
  f1score = 2*(precision*recall)/(precision+recall)
  return(f1score)
}

## To save classification metrics
## Fitxxx: metric for training subset. Testxxx: metric for testing subset
FitAccuracy = TestAccuracy <- array(NA,dim=c(SIM,4))  ## Accuracy Rate
FitPrecisionClass = TestPrecisionClass <- array(NA,dim=c(SIM,4,3))  ## Precision per class
FitRecallClass = TestRecallClass <- array(NA,dim=c(SIM,4,3))  ## Recall per class
FitF1ScoreClass = TestF1ScoreClass <- array(NA,dim=c(SIM,4,3))  ## F1-score per class
FitPrecisionMacroAve = TestPrecisionMacroAve <- array(NA,dim=c(SIM,4))  ## Precision Macro Average
FitRecallMacroAve = TestRecallMacroAve <- array(NA,dim=c(SIM,4))  ## Recall Macro Average
FitF1ScoreMacroAve = TestF1ScoreMacroAve <- array(NA,dim=c(SIM,4))  ## F1-score Macro Average
```

Model estimation

```
##-----------------------------------------------------
for(sim in 1:SIM){ ## BEGIN sim
##-----------------------------------------------------
my_fit = FIT[sim,]   ## training subset
my_test = TEST[sim,]   ## testing subset

## Training data subset
train1 <- trainc %>% filter(ID_fact%in%my_fit, rep==1)  ## repetition=1
train2 <- trainc %>% filter(ID_fact%in%my_fit, rep==2)  ## repetition=2
train3 <- trainc %>% filter(ID_fact%in%my_fit, rep==3)  ## repetition=3

Yc = train1$STATUS_fact   ## categorical response variable for training
n = length(Yc)
G = 3 # classes

## Testing data subset
test1 <- trainc %>% filter(ID_fact%in%my_test, rep==1)  ## repetition=1
test2 <- trainc %>% filter(ID_fact%in%my_test, rep==2)  ## repetition=2
test3 <- trainc %>% filter(ID_fact%in%my_test, rep==3)  ## repetition=3

Yc.new = test1$STATUS_fact   ## categorical response variable for testing
n.new = length(Yc.new)

## Delete variables which are not used
train1 <- train1 %>% select(-c(rep,ID_fact))
train2 <- train2 %>% select(-c(rep,ID_fact))
train3 <- train3 %>% select(-c(rep,ID_fact))
test1 <- test1 %>% select(-c(rep,ID_fact,STATUS_fact))
test2 <- test2 %>% select(-c(rep,ID_fact,STATUS_fact))
test3 <- test3 %>% select(-c(rep,ID_fact,STATUS_fact))

##-----------------------------------------------------
## Algorithm RSAM
## Replication-based stagewise additive modeling
##-----------------------------------------------------

## Algo1: Initialize the observation weights $w_i=1/n$, $i=1,...,n$
wi1 = rep(1/n,n)

## Algo2: BEGIN for replication j=1 to J do:

## REPLICATION j=1:
## Algo3: Fit a classifier $T(xj,z)$ to the training data using weights $wi$
mod1 <- rpart( STATUS_fact ~ . ,
               weights = wi1,
               data = train1)
## Predictions
pred1.vgam <- predict(mod1, newdata = train1)
pred1 <- apply(pred1.vgam,1,which.max)

## Algo4: Compute $err = \sum wi I[Y != T(xj,z)] / \sum wi$
```

```
err1 <- (sum(wi1*(Yc!=pred1))) / sum(wi1)
## Algo5: Compute $alpha = log (1-err)/err +log(G-1)$
alp1 <- log((1-err1)/err1) + log(G-1)
#alp1 <- ifelse(is.finite(alp1), alp1, log(G-1))
## Algo6: Set wi = wi* exp(alpha*I[Y \neq T(xj,z)])
wi2 = wi1*exp(alp1*(Yc!=pred1))
## Algo7: Re-normalize wi
wi2 = c(wi2/sum(wi2))

##-------------------------------------------------
## REPLICATION j=2:
## Algo3: Fit a classifier $T(xj,z)$ to the training data using weights $wi$
mod2 <- rpart( STATUS_fact ~ . ,
               weights = wi2,
               data = train2)
#summary(mod2)

### Predictions
pred2.vgam <- predict(mod2, newdata = train2)
pred2 <- apply(pred2.vgam,1,which.max)

## Algo4: Compute $err = \sum wi I[Y != T(xj,z)] / \sum wi$
err2 <- (sum(wi2*(Yc!=pred2))) / sum(wi2)
## Algo5: Compute $alpha = log (1-err)/err +log(G-1)$
alp2 <- log((1-err2)/err2) + log(G-1)
#alp2 <- ifelse(is.finite(alp2), alp2, log(G-1))
## Algo6: Set wi = wi* exp(alpha*I[Y \neq T(xj,z)])
wi3 = wi2*exp(alp2*(Yc!=pred2))
## Algo7: Re-normalize wi
wi3 = c(wi3/sum(wi3))

##-------------------------------------------------
## REPLICATION j=3:
## Algo3: Fit a classifier $T(xj,z)$ to the training data using weights $wi$
mod3 <- rpart( STATUS_fact ~ . ,
               weights = wi3,
               data = train3)
#summary(mod3)

### Predictions
pred3.vgam <- predict(mod3, newdata = train3)
pred3 <- apply(pred3.vgam,1,which.max)

## Algo4: Compute $err = \sum wi I[Y != T(xj,z)] / \sum wi$
err3 <- (sum(wi3*(Yc!=pred3))) / sum(wi3)
## Algo5: Compute $alpha = log (1-err)/err +log(G-1)$
alp3 <- log((1-err3)/err3) + log(G-1)
#alp3 <- ifelse(is.finite(alp3), alp3, log(G-1))
## Algo6: Set wi = wi* exp(alpha*I[Y \neq T(xj,z)])
wi4 = wi3*exp(alp3*(Yc!=pred3))
## Algo7: Re-normalize wi
wi4 = c(wi4/sum(wi4))
```

```
## Algo8: End for replication j=1 to J
##-------------------------------------------------

## Algo9: Output T*(x,z) = arg max_G \sum_j alpha*I[T(xj,z)=G]

pred = cbind(pred1,pred2,pred3)
alpha = c(alp1,alp2,alp3)

argclase = matrix(NA,n,3)
clase = rep(NA,n)
for(i in 1:n){
  argclase[i,1] = sum(alpha*(pred[i,]==1))
  argclase[i,2] = sum(alpha*(pred[i,]==2))
  argclase[i,3] = sum(alpha*(pred[i,]==3))
  clase[i] = which(argclase[i,]==max(argclase[i,]))
}
##-------------------------------------------------
## Predict new subjects for testing subsets

pred1.new.vgam <- predict(mod1, newdata = test1)
pred2.new.vgam <- predict(mod2, newdata = test2)
pred3.new.vgam <- predict(mod3, newdata = test3)
pred1.new <- apply(pred1.new.vgam,1,which.max)
pred2.new <- apply(pred2.new.vgam,1,which.max)
pred3.new <- apply(pred3.new.vgam,1,which.max)

pred.new = cbind(pred1.new,pred2.new,pred3.new)

argclase.new = matrix(NA,n.new,3)
clase.new = rep(NA,n.new)
for(i in 1:n.new){
  argclase.new[i,1] = sum(alpha*(pred.new[i,]==1))
  argclase.new[i,2] = sum(alpha*(pred.new[i,]==2))
  argclase.new[i,3] = sum(alpha*(pred.new[i,]==3))
  clase.new[i] = which(argclase.new[i,]==max(argclase.new[i,]))
}
##-------------------------------------------------
## End RSAM
##-------------------------------------------------
## Classification Metrics for models predicting nominal outcomes

## Accuracy Rate
FitAccuracy[sim,] = c(sum(Yc==pred1)/n,
                      sum(Yc==pred2)/n,
                      sum(Yc==pred3)/n,
                      sum(Yc==clase)/n)

TestAccuracy[sim,] = c(sum(Yc.new==pred1.new)/n.new,
                       sum(Yc.new==pred2.new)/n.new,
                       sum(Yc.new==pred3.new)/n.new,
                       sum(Yc.new==clase.new)/n.new)

## Precision
```

```
for(cate in 1:3){
  FitPrecisionClass[sim,1, cate] = fn_precision_class(Yc, pred1, cate)
  FitPrecisionClass[sim,2, cate] = fn_precision_class(Yc, pred2, cate)
  FitPrecisionClass[sim,3, cate] = fn_precision_class(Yc, pred3, cate)
  FitPrecisionClass[sim,4, cate] = fn_precision_class(Yc, clase, cate)

  TestPrecisionClass[sim,1, cate] = fn_precision_class(Yc.new, pred1.new, cate)
  TestPrecisionClass[sim,2, cate] = fn_precision_class(Yc.new, pred2.new, cate)
  TestPrecisionClass[sim,3, cate] = fn_precision_class(Yc.new, pred3.new, cate)
  TestPrecisionClass[sim,4, cate] = fn_precision_class(Yc.new, clase.new, cate)
}
for(rep in 1:4){
  FitPrecisionMacroAve[sim, rep] = mean(FitPrecisionClass[sim, rep,])
  TestPrecisionMacroAve[sim,rep] = mean(TestPrecisionClass[sim,rep,])
}

## Recall
for(cate in 1:3){
  FitRecallClass[sim,1, cate] = fn_recall_class(Yc, pred1, cate)
  FitRecallClass[sim,2, cate] = fn_recall_class(Yc, pred2, cate)
  FitRecallClass[sim,3, cate] = fn_recall_class(Yc, pred3, cate)
  FitRecallClass[sim,4, cate] = fn_recall_class(Yc, clase, cate)

  TestRecallClass[sim,1, cate] = fn_recall_class(Yc.new, pred1.new, cate)
  TestRecallClass[sim,2, cate] = fn_recall_class(Yc.new, pred2.new, cate)
  TestRecallClass[sim,3, cate] = fn_recall_class(Yc.new, pred3.new, cate)
  TestRecallClass[sim,4, cate] = fn_recall_class(Yc.new, clase.new, cate)
}
for(rep in 1:4){
  FitRecallMacroAve[sim, rep] = mean(FitRecallClass[sim, rep,])
  TestRecallMacroAve[sim,rep] = mean(TestRecallClass[sim,rep,])
}

## F1-Score
for(cate in 1:3){
  FitF1ScoreClass[sim,1, cate]= fn_f1score_class(Yc, pred1, cate)
  FitF1ScoreClass[sim,2, cate]= fn_f1score_class(Yc, pred2, cate)
  FitF1ScoreClass[sim,3, cate]= fn_f1score_class(Yc, pred3, cate)
  FitF1ScoreClass[sim,4, cate]= fn_f1score_class(Yc, clase, cate)

  TestF1ScoreClass[sim,1, cate] = fn_f1score_class(Yc.new, pred1.new, cate)
  TestF1ScoreClass[sim,2, cate] = fn_f1score_class(Yc.new, pred2.new, cate)
  TestF1ScoreClass[sim,3, cate] = fn_f1score_class(Yc.new, pred3.new, cate)
  TestF1ScoreClass[sim,4, cate] = fn_f1score_class(Yc.new, clase.new, cate)
}
for(rep in 1:4){
  FitF1ScoreMacroAve[sim, rep] = mean(FitF1ScoreClass[sim, rep,])
  TestF1ScoreMacroAve[sim,rep] = mean(TestF1ScoreClass[sim,rep,])
}
##------------------------------------------------
}## END sim
##------------------------------------------------
```

Results

Accuracy Rate

```
columna = c("rep1","rep2","rep3","ensemble")
renglon = c("fit_mean","fit_sd","test_mean","test_sd")

summary(FitAccuracy)
```

```
##        V1               V2               V3               V4
##  Min.   :0.7733   Min.   :0.7511   Min.   :0.6978   Min.   :0.9333
##  1st Qu.:0.8167   1st Qu.:0.7911   1st Qu.:0.7733   1st Qu.:0.9689
##  Median :0.8289   Median :0.8111   Median :0.7822   Median :0.9733
##  Mean   :0.8289   Mean   :0.8106   Mean   :0.7859   Mean   :0.9742
##  3rd Qu.:0.8400   3rd Qu.:0.8278   3rd Qu.:0.8000   3rd Qu.:0.9822
##  Max.   :0.8844   Max.   :0.8667   Max.   :0.8533   Max.   :0.9956
```

```
apply(FitAccuracy,2,"sd")
```

```
## [1] 0.02060078 0.02586741 0.02552264 0.01131794
```

```
summary(TestAccuracy)
```

```
##        V1               V2               V3               V4
##  Min.   :0.5200   Min.   :0.5333   Min.   :0.5067   Min.   :0.6400
##  1st Qu.:0.6267   1st Qu.:0.6667   1st Qu.:0.6400   1st Qu.:0.7467
##  Median :0.6533   Median :0.6933   Median :0.6800   Median :0.7867
##  Mean   :0.6568   Mean   :0.6899   Mean   :0.6764   Mean   :0.7863
##  3rd Qu.:0.6833   3rd Qu.:0.7200   3rd Qu.:0.7200   3rd Qu.:0.8267
##  Max.   :0.7733   Max.   :0.8000   Max.   :0.8533   Max.   :0.8800
```

```
apply(TestAccuracy,2,"sd")
```

```
## [1] 0.05223172 0.04998298 0.06108205 0.05205384
```

```
RESaccuracy <- rbind(apply(FitAccuracy,2,"mean"), apply(FitAccuracy,2,"sd"),
                     apply(TestAccuracy,2,"mean"),apply(TestAccuracy,2,"sd"))
colnames(RESaccuracy) = columna
rownames(RESaccuracy) = renglon
write.csv(RESaccuracy, file=paste0(archivo,"_accuracy",".csv"))
```

Precision Macro Average

```
summary(FitPrecisionMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.7940   Min.   :0.7511   Min.   :0.7087   Min.   :0.9364
##  1st Qu.:0.8201   1st Qu.:0.7991   1st Qu.:0.7781   1st Qu.:0.9691
##  Median :0.8332   Median :0.8148   Median :0.7910   Median :0.9738
##  Mean   :0.8339   Mean   :0.8165   Mean   :0.7923   Mean   :0.9746
##  3rd Qu.:0.8444   3rd Qu.:0.8334   3rd Qu.:0.8087   3rd Qu.:0.9823
##  Max.   :0.8856   Max.   :0.8685   Max.   :0.8540   Max.   :0.9956
```

```
apply(FitPrecisionMacroAve,2,"sd")
```

```
## [1] 0.01905830 0.02450931 0.02455607 0.01103619
```

```
summary(TestPrecisionMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.5104   Min.   :0.5868   Min.   :0.5176   Min.   :0.6653
##  1st Qu.:0.6312   1st Qu.:0.6713   1st Qu.:0.6476   1st Qu.:0.7603
##  Median :0.6636   Median :0.6947   Median :0.6967   Median :0.7909
##  Mean   :0.6636   Mean   :0.7023   Mean   :0.6873   Mean   :0.7925
##  3rd Qu.:0.6970   3rd Qu.:0.7390   3rd Qu.:0.7307   3rd Qu.:0.8343
##  Max.   :0.7810   Max.   :0.8203   Max.   :0.8645   Max.   :0.8870
```

```
apply(TestPrecisionMacroAve,2,"sd")
```

```
## [1] 0.05233346 0.04934068 0.06334102 0.05070212
```

```
RESprecision <- rbind(apply(FitPrecisionMacroAve,2,"mean"), apply(FitPrecisionMacroAve,2,"sd"),
                      apply(TestPrecisionMacroAve,2,"mean"),apply(TestPrecisionMacroAve,2,"sd"))
colnames(RESprecision) = columna
rownames(RESprecision) = renglon
write.csv(RESprecision, file=paste0(archivo,"_precision",".csv"))
```

Recall Macro Average

```
summary(FitRecallMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.7733   Min.   :0.7511   Min.   :0.6978   Min.   :0.9333
##  1st Qu.:0.8167   1st Qu.:0.7911   1st Qu.:0.7733   1st Qu.:0.9689
##  Median :0.8289   Median :0.8111   Median :0.7822   Median :0.9733
##  Mean   :0.8289   Mean   :0.8106   Mean   :0.7859   Mean   :0.9742
##  3rd Qu.:0.8400   3rd Qu.:0.8278   3rd Qu.:0.8000   3rd Qu.:0.9822
##  Max.   :0.8844   Max.   :0.8667   Max.   :0.8533   Max.   :0.9956
```

```
apply(FitRecallMacroAve,2,"sd")
```

```
## [1] 0.02060078 0.02586741 0.02552264 0.01131794
```

```
summary(TestRecallMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.5200   Min.   :0.5333   Min.   :0.5067   Min.   :0.6400
##  1st Qu.:0.6267   1st Qu.:0.6667   1st Qu.:0.6400   1st Qu.:0.7467
##  Median :0.6533   Median :0.6933   Median :0.6800   Median :0.7867
##  Mean   :0.6568   Mean   :0.6899   Mean   :0.6764   Mean   :0.7863
##  3rd Qu.:0.6833   3rd Qu.:0.7200   3rd Qu.:0.7200   3rd Qu.:0.8267
##  Max.   :0.7733   Max.   :0.8000   Max.   :0.8533   Max.   :0.8800
```

```
apply(TestRecallMacroAve,2,"sd")
```

```
## [1] 0.05223172 0.04998298 0.06108205 0.05205384
```

```
RESrecall <- rbind(apply(FitRecallMacroAve,2,"mean"), apply(FitRecallMacroAve,2,"sd"),
                   apply(TestRecallMacroAve,2,"mean"),apply(TestRecallMacroAve,2,"sd"))
colnames(RESrecall) = columna
rownames(RESrecall) = renglon
write.csv(RESrecall, file=paste0(archivo,"_recall",".csv"))
```

F1-Score Macro Average

```
summary(FitF1ScoreMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.7703   Min.   :0.7497   Min.   :0.6919   Min.   :0.9337
##  1st Qu.:0.8162   1st Qu.:0.7912   1st Qu.:0.7724   1st Qu.:0.9688
##  Median :0.8286   Median :0.8092   Median :0.7823   Median :0.9734
##  Mean   :0.8282   Mean   :0.8101   Mean   :0.7850   Mean   :0.9742
##  3rd Qu.:0.8390   3rd Qu.:0.8278   3rd Qu.:0.7993   3rd Qu.:0.9822
##  Max.   :0.8847   Max.   :0.8661   Max.   :0.8514   Max.   :0.9956
```

```
apply(FitF1ScoreMacroAve,2,"sd")
```

```
## [1] 0.02093468 0.02588865 0.02608635 0.01131882
```

```
summary(TestF1ScoreMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.5094   Min.   :0.5396   Min.   :0.5082   Min.   :0.6333
##  1st Qu.:0.6239   1st Qu.:0.6589   1st Qu.:0.6364   1st Qu.:0.7475
##  Median :0.6538   Median :0.6875   Median :0.6803   Median :0.7859
##  Mean   :0.6546   Mean   :0.6872   Mean   :0.6729   Mean   :0.7854
##  3rd Qu.:0.6829   3rd Qu.:0.7203   3rd Qu.:0.7127   3rd Qu.:0.8271
##  Max.   :0.7737   Max.   :0.7981   Max.   :0.8512   Max.   :0.8817
```

```
apply(TestF1ScoreMacroAve,2,"sd")
```

```
## [1] 0.05304397 0.05082915 0.06254338 0.05260658
```

```
RESf1score <- rbind(apply(FitF1ScoreMacroAve,2,"mean"), apply(FitF1ScoreMacroAve,2,"sd"),
                    apply(TestF1ScoreMacroAve,2,"mean"),apply(TestF1ScoreMacroAve,2,"sd"))
colnames(RESf1score) = columna
rownames(RESf1score) = renglon
write.csv(RESf1score, file=paste0(archivo,"_f1score",".csv"))
```