Random Forest (RF)

Replication-based stagewise additive modeling (RSAM)

EGG data-based experiments

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.

```
library(tidyverse)
library(randomForest)
## change the address where the file will be saved
address = "~/Documents/GitHub/"
setwd("~/Documents/GitHub/")
```

EGG data-based experiments

```
## Comment or uncomment the options: EGG-a, EGG-i, EGG-u
```

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

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

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

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

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

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

```r
dim(datos2)
```

```
[1] 675  36
```

```r
summary(datos2)
```

```
    ID_fact       status_fact      SEX             JITTER      
 Min.   :  1.0   Min.   :0    Min.   :0.0000   Min.   :  0.00  
 1st Qu.:169.5   1st Qu.:0    1st Qu.:0.0000   1st Qu.:  0.45  
 Median :338.0   Median :1    Median :0.0000   Median :  1.06  
 Mean   :338.0   Mean   :1    Mean   :0.4133   Mean   : 13.84  
 3rd Qu.:506.5   3rd Qu.:2    3rd Qu.:1.0000   3rd Qu.: 17.95  
 Max.   :675.0   Max.   :2    Max.   :1.0000   Max.   :273.97  
    SHIMMER             CPP             D2              FZCF        
 Min.   :0.00000   Min.   :12.04   Min.   : 2.000   Min.   :   5.00  
 1st Qu.:0.03000   1st Qu.:18.02   1st Qu.: 3.625   1st Qu.:  11.00  
 Median :0.05000   Median :20.86   Median : 4.410   Median :  16.00  
 Mean   :0.06012   Mean   :21.36   Mean   : 4.738   Mean   :  42.47  
 3rd Qu.:0.08000   3rd Qu.:23.95   3rd Qu.: 5.445   3rd Qu.:  23.00  
 Max.   :0.38000   Max.   :34.19   Max.   :18.380   Max.   :5280.00  
      GNE              HNR           HURST              LZ       
 Min.   :0.4100   Min.   :-3.94   Min.   :0.1000   Min.   : 19.0  
 1st Qu.:0.6350   1st Qu.:20.15   1st Qu.:0.6150   1st Qu.: 37.0  
 Median :0.8000   Median :23.48   Median :0.8700   Median : 50.0  
 Mean   :0.9465   Mean   :22.34   Mean   :0.8806   Mean   : 54.2  
 3rd Qu.:1.0950   3rd Qu.:26.51   3rd Qu.:1.1350   3rd Qu.: 65.5  
 Max.   :5.0900   Max.   :33.91   Max.   :1.7700   Max.   :279.0  
     MFCC0             MFCC1            MFCC2             MFCC3        
 Min.   :-2.8800   Min.   :-19.06   Min.   :-30.610   Min.   :-45.920  
 1st Qu.:-1.1200   1st Qu.: 11.15   1st Qu.: -0.635   1st Qu.:-21.495  
 Median :-0.5300   Median : 18.10   Median : 10.670   Median : -7.850  
 Mean   :-0.5224   Mean   : 16.26   Mean   :  9.860   Mean   : -8.324  
 3rd Qu.: 0.1200   3rd Qu.: 22.50   3rd Qu.: 19.775   3rd Qu.:  3.180  
 Max.   : 2.0500   Max.   : 32.80   Max.   : 46.420   Max.   : 48.690  
     MFCC4             MFCC5             MFCC6             MFCC7        
 Min.   :-57.250   Min.   :-43.400   Min.   :-43.040   Min.   :-41.830  
 1st Qu.:-20.330   1st Qu.:-14.910   1st Qu.:-15.365   1st Qu.:-16.115  
 Median :-10.640   Median : -7.710   Median : -8.610   Median : -8.020  
 Mean   :-11.226   Mean   : -7.712   Mean   : -8.940   Mean   : -8.249  
 3rd Qu.: -2.105   3rd Qu.:  0.945   3rd Qu.: -2.035   3rd Qu.: -0.215  
 Max.   : 25.070   Max.   : 34.400   Max.   : 26.060   Max.   : 36.600  
     MFCC8             MFCC9            MFCC10            MFCC11       
 Min.   :-51.090   Min.   :-47.06   Min.   :-39.590   Min.   :-39.150  
 1st Qu.:-15.975   1st Qu.:-13.44   1st Qu.:-12.060   1st Qu.:-12.575  
 Median : -7.810   Median : -6.26   Median : -4.420   Median : -4.380  
 Mean   : -7.906   Mean   : -5.64   Mean   : -4.325   Mean   : -4.466  
 3rd Qu.: -0.240   3rd Qu.:  1.71   3rd Qu.:  2.080   3rd Qu.:  2.885  
 Max.   : 36.520   Max.   : 40.73   Max.   : 42.350   Max.   : 35.350  
     MFCC12         PERMUTATION         PPE            SHANNON     
 Min.   :-37.200   Min.   :1.110   Min.   :0.0000   Min.   :11.92  
 1st Qu.:-11.775   1st Qu.:1.440   1st Qu.:0.5300   1st Qu.:12.16  
 Median : -5.020   Median :1.570   Median :0.5500   Median :12.19  
 Mean   : -4.221   Mean   :1.642   Mean   :0.5315   Mean   :12.18  
 3rd Qu.:  2.625   3rd Qu.:1.780   3rd Qu.:0.5700   3rd Qu.:12.21  
 Max.   : 29.550   Max.   :2.580   Max.   :0.5700   Max.   :12.26  
      ZCR          energyentropy   spectralcentroid spectralspread  
 Min.   :0.01000   Min.   :2.500   Min.   :0.0700   Min.   :0.1200  
 1st Qu.:0.02000   1st Qu.:3.260   1st Qu.:0.1100   1st Qu.:0.1600  
 Median :0.03000   Median :3.310   Median :0.1200   Median :0.1800  
 Mean   :0.03846   Mean   :3.269   Mean   :0.1206   Mean   :0.1798  
 3rd Qu.:0.05000   3rd Qu.:3.320   3rd Qu.:0.1300   3rd Qu.:0.1900  
 Max.   :0.20000   Max.   :3.320   Max.   :0.3300   Max.   :0.3300  
 spectralentropy  spectralrolloff        RPDE             rep   
 Min.   :0.0000   Min.   :0.01000   Min.   :0.0100   Min.   :1  
 1st Qu.:0.0500   1st Qu.:0.04000   1st Qu.:0.2000   1st Qu.:1  
 Median :0.1100   Median :0.05000   Median :0.2800   Median :2  
 Mean   :0.1832   Mean   :0.05613   Mean   :0.3194   Mean   :2  
 3rd Qu.:0.2400   3rd Qu.:0.07000   3rd Qu.:0.3900   3rd Qu.:3  
 Max.   :1.6400   Max.   :0.37000   Max.   :0.9000   Max.   :3  
```

```
head(datos2)
```

```
  ID_fact status_fact SEX JITTER SHIMMER   CPP   D2 FZCF  GNE   HNR HURST LZ
1       1           0   0   0.21    0.02 28.48 4.35   25 0.63 27.12  1.20 32
2       2           0   0   0.43    0.06 22.18 3.23   31 0.58 18.23  1.38 44
3       3           0   0   0.46    0.03 24.91 5.24   27 0.60 24.93  1.30 33
4       4           0   0   0.49    0.02 31.64 3.14   24 0.57 25.71  1.17 34
5       5           0   0  11.39    0.09 24.74 2.16   40 0.43 15.64  1.51 39
6       6           0   0   0.33    0.03 29.29 3.27   33 0.47 24.57  1.40 33
  MFCC0 MFCC1 MFCC2 MFCC3 MFCC4 MFCC5 MFCC6 MFCC7 MFCC8 MFCC9 MFCC10 MFCC11
1 -0.41  8.69 -0.89  0.03  3.66 -0.03 -0.20 -3.42 -4.11 -6.53  -8.78  -9.64
2 -1.47  3.10 15.86  5.73  9.91  4.40  4.72  0.60  2.27 -2.16  -3.27  -3.26
3 -0.92  7.38 11.85  2.93  2.55  2.91  3.56 -1.11 -1.74 -4.55  -4.97  -7.97
4 -0.52  8.81 -2.68  2.83 -0.76 -0.60 -2.30 -5.78 -4.10 -7.55  -7.57  -8.92
5 -2.06 11.23  9.58  5.36  7.67  2.17  3.94  4.12  4.38  0.62   0.97   1.00
6 -1.39 14.83  3.50 -2.30  9.86  2.90  1.00  6.24 -1.52  1.44  -0.26  -1.19
  MFCC12 PERMUTATION  PPE SHANNON  ZCR energyentropy spectralcentroid
1 -11.13        2.02 0.55   12.19 0.02          3.31             0.13
2  -3.81        2.26 0.48   12.20 0.02          3.27             0.12
3  -5.99        1.99 0.55   12.21 0.02          3.30             0.12
4  -8.74        1.68 0.55   12.17 0.02          3.31             0.12
5  -0.09        1.84 0.44   12.19 0.01          3.19             0.09
6  -1.26        1.64 0.53   12.17 0.01          3.28             0.11
  spectralspread spectralentropy spectralrolloff RPDE rep
1           0.19            0.11            0.02 0.21   1
2           0.21            0.06            0.02 0.50   2
3           0.21            0.06            0.02 0.32   3
4           0.18            0.13            0.03 0.32   1
5           0.17            0.03            0.01 0.58   2
6           0.18            0.08            0.02 0.37   3
```

Re-Scale explanatory variables

```
## Scale the variables
datos2 <- as.data.frame(datos2)
datos2$STATUS_fact = as.factor(as.numeric(factor(datos2$status_fact)))

table(datos2$STATUS_fact)
```

```

  1   2   3 
225 225 225 
```

```
datos <- transform(datos2,
sJITTER= scale(JITTER), sSHIMMER= scale(SHIMMER), sCPP= scale(CPP),
sD2= scale(D2), sFZCF= scale(FZCF), sGNE= scale(GNE),
sHNR= scale(HNR), sHURST= scale(HURST), sLZ= scale(LZ),
sMFCC0= scale(MFCC0),
sMFCC1= scale(MFCC1), sMFCC2= scale(MFCC2), sMFCC3= scale(MFCC3),
sMFCC4= scale(MFCC4), sMFCC5= scale(MFCC5), sMFCC6= scale(MFCC6),
sMFCC7= scale(MFCC7), sMFCC8= scale(MFCC8), sMFCC9= scale(MFCC9),
sMFCC10= scale(MFCC10), sMFCC11= scale(MFCC11), sMFCC12= scale(MFCC12),
sPERMUTATION= scale(PERMUTATION), sPPE= scale(PPE), sSHANNON= scale(SHANNON),
sZCR= scale(ZCR),
senergyentropy= scale(energyentropy), sspectralcentroid= scale(spectralcentroid),
sspectralspread= scale(spectralspread), sspectralentropy= scale(spectralentropy),
sspectralrolloff= scale(spectralrolloff), sRPDE= scale(RPDE))

datos$ID_fact = rep(1:225,each=3)

dim(datos)
```

```
[1] 675  69
```

```
## data set
trainc <- datos %>% select(
sJITTER, sSHIMMER, sCPP, sD2, sFZCF,
sGNE, sHNR, sHURST, sLZ, sMFCC0,
sMFCC1, sMFCC2, sMFCC3, sMFCC4, sMFCC5,
sMFCC6, sMFCC7, sMFCC8, sMFCC9, sMFCC10,
sMFCC11, sMFCC12,
sPERMUTATION, sPPE, sSHANNON, sZCR,
senergyentropy, sspectralcentroid, sspectralspread,
sspectralentropy, sspectralrolloff, sRPDE,
STATUS_fact,SEX, rep,ID_fact)
```

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 = 225   ## sample size
Nfit = 168   ## sample size for training subset
Ntest = 57   ## sample size for testing subset
Ncat = 75   ## sample size per category
Ncatfit = 56   ## training per category
Ncattest = 19   ## 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

```r
## 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

```r
##--------------------------------------------------
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(SEX,rep,ID_fact,STATUS_fact))
train2 <- train2 %>% select(-c(SEX,rep,ID_fact,STATUS_fact))
train3 <- train3 %>% select(-c(SEX,rep,ID_fact,STATUS_fact))
test1 <- test1 %>% select(-c(SEX,rep,ID_fact,STATUS_fact))
test2 <- test2 %>% select(-c(SEX,rep,ID_fact,STATUS_fact))
test3 <- test3 %>% select(-c(SEX,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 <- randomForest(
  x = train1 ,
  y = Yc  ,
  weights = wi1,
  ntree = 500,
  xtest = test1)
## summary(mod1)
## Predictions
pred1.vgam <- mod1$votes
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 <- randomForest(
  x = train2 ,
  y = Yc  ,
  weights = wi2,
  ntree = 500,
  xtest = test2)
## summary(mod2)

## Predictions
pred2.vgam <- mod2$votes
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 <- randomForest(
  x = train3 ,
  y = Yc  ,
  weights = wi1,
  ntree = 500,
  xtest = test3)
## summary(mod3)

## Predictions
pred3.vgam <- mod3$votes
for(i in 1:n){ if(is.na(pred3.vgam[i,])){
  pred3.vgam[i,] =1/3
} }
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 <- mod1$test$votes
pred2.new.vgam <- mod2$test$votes
pred3.new.vgam <- mod3$test$votes
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)
```

```r
  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.   :0.5060   Min.   :0.4345   Min.   :0.5536   Min.   :0.5417
##  1st Qu.:0.5595   1st Qu.:0.5223   1st Qu.:0.6131   1st Qu.:0.5878
##  Median :0.5833   Median :0.5387   Median :0.6310   Median :0.6131
##  Mean   :0.5816   Mean   :0.5378   Mean   :0.6304   Mean   :0.6095
##  3rd Qu.:0.6012   3rd Qu.:0.5536   3rd Qu.:0.6503   3rd Qu.:0.6310
##  Max.   :0.6548   Max.   :0.6250   Max.   :0.6905   Max.   :0.6667
```

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

```
## [1] 0.02815268 0.03304277 0.02879123 0.02719044
```

```
summary(TestAccuracy)
```

```
##        V1               V2               V3               V4
##  Min.   :0.4912   Min.   :0.4035   Min.   :0.5088   Min.   :0.4912
##  1st Qu.:0.5439   1st Qu.:0.5263   1st Qu.:0.5965   1st Qu.:0.5789
##  Median :0.5965   Median :0.5614   Median :0.6491   Median :0.6140
##  Mean   :0.5960   Mean   :0.5644   Mean   :0.6389   Mean   :0.6167
##  3rd Qu.:0.6316   3rd Qu.:0.6140   3rd Qu.:0.6667   3rd Qu.:0.6667
##  Max.   :0.7018   Max.   :0.7193   Max.   :0.7895   Max.   :0.7368
```

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

```
## [1] 0.05527954 0.06416077 0.05593508 0.05758891
```

```
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

```r
summary(FitPrecisionMacroAve)
```

```
##        V1               V2               V3               V4        
##  Min.   :0.5230   Min.   :0.4282   Min.   :0.5581   Min.   :0.5532  
##  1st Qu.:0.5674   1st Qu.:0.5164   1st Qu.:0.6195   1st Qu.:0.5967  
##  Median :0.5917   Median :0.5310   Median :0.6418   Median :0.6202  
##  Mean   :0.5883   Mean   :0.5337   Mean   :0.6395   Mean   :0.6185  
##  3rd Qu.:0.6066   3rd Qu.:0.5506   3rd Qu.:0.6603   3rd Qu.:0.6383  
##  Max.   :0.6584   Max.   :0.6405   Max.   :0.6975   Max.   :0.6752
```

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

```
## [1] 0.02818108 0.03660251 0.02901607 0.02797019
```

```r
summary(TestPrecisionMacroAve)
```

```
##        V1               V2               V3               V4        
##  Min.   :0.5028   Min.   :0.3695   Min.   :0.5143   Min.   :0.5054  
##  1st Qu.:0.5664   1st Qu.:0.5174   1st Qu.:0.6165   1st Qu.:0.5897  
##  Median :0.6166   Median :0.5645   Median :0.6568   Median :0.6344  
##  Mean   :0.6101   Mean   :0.5641   Mean   :0.6549   Mean   :0.6323  
##  3rd Qu.:0.6496   3rd Qu.:0.6109   3rd Qu.:0.6815   3rd Qu.:0.6816  
##  Max.   :0.7233   Max.   :0.7177   Max.   :0.8074   Max.   :0.7365
```

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

```
## [1] 0.05514761 0.06818061 0.05436537 0.05478997
```

```r
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.   :0.5060   Min.   :0.4345   Min.   :0.5536   Min.   :0.5417
##  1st Qu.:0.5595   1st Qu.:0.5223   1st Qu.:0.6131   1st Qu.:0.5878
##  Median :0.5833   Median :0.5387   Median :0.6310   Median :0.6131
##  Mean   :0.5816   Mean   :0.5378   Mean   :0.6304   Mean   :0.6095
##  3rd Qu.:0.6012   3rd Qu.:0.5536   3rd Qu.:0.6503   3rd Qu.:0.6310
##  Max.   :0.6548   Max.   :0.6250   Max.   :0.6905   Max.   :0.6667
```

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

```
## [1] 0.02815268 0.03304277 0.02879123 0.02719044
```

```
summary(TestRecallMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.4912   Min.   :0.4035   Min.   :0.5088   Min.   :0.4912
##  1st Qu.:0.5439   1st Qu.:0.5263   1st Qu.:0.5965   1st Qu.:0.5789
##  Median :0.5965   Median :0.5614   Median :0.6491   Median :0.6140
##  Mean   :0.5960   Mean   :0.5644   Mean   :0.6389   Mean   :0.6167
##  3rd Qu.:0.6316   3rd Qu.:0.6140   3rd Qu.:0.6667   3rd Qu.:0.6667
##  Max.   :0.7018   Max.   :0.7193   Max.   :0.7895   Max.   :0.7368
```

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

```
## [1] 0.05527954 0.06416077 0.05593508 0.05758891
```

```
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

```r
summary(FitF1ScoreMacroAve)
```

```
##        V1               V2               V3               V4        
##  Min.   :0.5107   Min.   :0.4305   Min.   :0.5552   Min.   :0.5449  
##  1st Qu.:0.5619   1st Qu.:0.5173   1st Qu.:0.6148   1st Qu.:0.5903  
##  Median :0.5847   Median :0.5332   Median :0.6340   Median :0.6165  
##  Mean   :0.5827   Mean   :0.5339   Mean   :0.6327   Mean   :0.6118  
##  3rd Qu.:0.6010   3rd Qu.:0.5504   3rd Qu.:0.6535   3rd Qu.:0.6328  
##  Max.   :0.6549   Max.   :0.6289   Max.   :0.6919   Max.   :0.6684
```

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

```
## [1] 0.02801126 0.03492838 0.02858255 0.02708565
```

```r
summary(TestF1ScoreMacroAve)
```

```
##        V1               V2               V3               V4        
##  Min.   :0.4631   Min.   :0.3831   Min.   :0.5053   Min.   :0.4891  
##  1st Qu.:0.5455   1st Qu.:0.5167   1st Qu.:0.5990   1st Qu.:0.5807  
##  Median :0.5992   Median :0.5601   Median :0.6466   Median :0.6179  
##  Mean   :0.5949   Mean   :0.5586   Mean   :0.6401   Mean   :0.6173  
##  3rd Qu.:0.6345   3rd Qu.:0.6029   3rd Qu.:0.6682   3rd Qu.:0.6633  
##  Max.   :0.7037   Max.   :0.7182   Max.   :0.7913   Max.   :0.7363
```

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

```
## [1] 0.05627791 0.06375982 0.05539822 0.05690040
```

```r
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"))
```