

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_Saarbrucken_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_Saarbrucken_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_Saarbrucken_egg_u"
```

```
dim(datos2)
```

```
[1] 675 36
```

```
summary(datos2)
```

ID_fact	status_fact	SEX	JITTER
Min. : 1.0	Min. :0	Min. :0.0000	Min. : 0.12
1st Qu.:169.5	1st Qu.:0	1st Qu.:0.0000	1st Qu.: 0.48
Median :338.0	Median :1	Median :0.0000	Median : 2.54
Mean :338.0	Mean :1	Mean :0.4133	Mean : 16.37
3rd Qu.:506.5	3rd Qu.:2	3rd Qu.:1.0000	3rd Qu.: 23.20
Max. :675.0	Max. :2	Max. :1.0000	Max. :385.73
SHIMMER	CPP	D2	FZCF
Min. :0.00000	Min. :11.69	Min. : -2.000	Min. : 3.00
1st Qu.:0.02000	1st Qu.:20.48	1st Qu.: 4.170	1st Qu.: 11.00
Median :0.03000	Median :24.04	Median : 5.160	Median : 16.00
Mean :0.04761	Mean :23.76	Mean : 5.487	Mean : 42.12
3rd Qu.:0.06000	3rd Qu.:27.11	3rd Qu.: 6.300	3rd Qu.: 23.00
Max. :0.32000	Max. :35.21	Max. :22.570	Max. :5280.00
GENE	HNHR	HURST	LZ
Min. :0.3600	Min. : -3.47	Min. :0.1100	Min. : 21.00
1st Qu.:0.6500	1st Qu.:18.54	1st Qu.:0.6200	1st Qu.: 36.00
Median :0.8300	Median :22.73	Median :0.8800	Median : 48.00
Mean :0.9723	Mean :21.57	Mean :0.9036	Mean : 59.29
3rd Qu.:1.1500	3rd Qu.:25.78	3rd Qu.:1.1700	3rd Qu.: 71.00
Max. :6.6900	Max. :33.35	Max. :1.7800	Max. :357.00
MFCC0	MFCC1	MFCC2	MFCC3
Min. : -2.75000	Min. : -33.1300	Min. : -22.68	Min. : -51.850
1st Qu.: -0.58000	1st Qu.: -5.9300	1st Qu.: 1.59	1st Qu.: -4.685
Median : 0.04000	Median : -0.9000	Median : 13.69	Median : 9.290
Mean : 0.01567	Mean : -0.9855	Mean : 12.40	Mean : 8.217
3rd Qu.: 0.59500	3rd Qu.: 4.1400	3rd Qu.: 22.55	3rd Qu.: 25.480
Max. : 2.51000	Max. : 18.3700	Max. : 41.77	Max. : 56.000
MFCC4	MFCC5	MFCC6	MFCC7
Min. : -65.180	Min. : -51.930	Min. : -53.77	Min. : -49.750
1st Qu.: -29.390	1st Qu.: -18.895	1st Qu.: -23.14	1st Qu.: -18.130
Median : -11.470	Median : -7.340	Median : -11.97	Median : -7.950
Mean : -11.149	Mean : -8.123	Mean : -12.41	Mean : -8.902
3rd Qu.: 4.905	3rd Qu.: 1.370	3rd Qu.: -1.29	3rd Qu.: 1.130
Max. : 46.890	Max. : 56.520	Max. : 21.69	Max. : 32.570
MFCC8	MFCC9	MFCC10	MFCC11
Min. : -49.410	Min. : -47.060	Min. : -40.290	Min. : -50.030
1st Qu.: -17.460	1st Qu.: -14.730	1st Qu.: -14.505	1st Qu.: -12.990
Median : -9.870	Median : -7.730	Median : -5.940	Median : -5.050
Mean : -9.593	Mean : -6.749	Mean : -5.744	Mean : -4.704
3rd Qu.: -1.320	3rd Qu.: 1.615	3rd Qu.: 2.100	3rd Qu.: 2.535
Max. : 36.430	Max. : 38.810	Max. : 36.160	Max. : 34.840
MFCC12	PERMUTATION	PPE	SHANNON
Min. : -33.830	Min. :1.160	Min. :0.1800	Min. :11.89
1st Qu.: -11.390	1st Qu.:1.830	1st Qu.:0.5300	1st Qu.:12.16
Median : -3.830	Median :2.090	Median :0.5500	Median :12.20
Mean : -3.119	Mean :2.037	Mean :0.5308	Mean :12.19

3rd Qu.: 3.700	3rd Qu.:2.295	3rd Qu.:0.5700	3rd Qu.:12.22
Max. : 39.690	Max. :2.580	Max. :0.5700	Max. :12.26
ZCR	energyentropy	spectralcentroid	spectralspread
Min. :0.01000	Min. :2.630	Min. :0.0800	Min. :0.1400
1st Qu.:0.03000	1st Qu.:3.280	1st Qu.:0.1400	1st Qu.:0.2000
Median :0.04000	Median :3.310	Median :0.1700	Median :0.2200
Mean :0.04539	Mean :3.279	Mean :0.1767	Mean :0.2285
3rd Qu.:0.05000	3rd Qu.:3.320	3rd Qu.:0.2000	3rd Qu.:0.2500
Max. :0.37000	Max. :3.320	Max. :0.4600	Max. :0.3300
spectralentropy	spectralrolloff	RPDE	rep
Min. :0.0200	Min. :0.0100	Min. :0.0100	Min. :1
1st Qu.:0.0900	1st Qu.:0.0300	1st Qu.:0.2100	1st Qu.:1
Median :0.1500	Median :0.0400	Median :0.3000	Median :2
Mean :0.2261	Mean :0.0608	Mean :0.3278	Mean :2
3rd Qu.:0.2700	3rd Qu.:0.0600	3rd Qu.:0.4100	3rd Qu.:3
Max. :2.2500	Max. :0.7800	Max. :0.8700	Max. :3

```
head(datos2)
```

	ID_fact	status_fact	SEX	JITTER	SHIMMER	CPP	D2	FZCF	GNE	HNR	HURST	LZ	
1	1		0	0	0.39	0.02	30.04	3.99	28	0.54	26.69	1.30	32
2	2		0	0	0.32	0.07	23.02	3.18	44	0.47	18.56	1.58	33
3	3		0	0	0.23	0.03	24.09	4.15	47	0.45	24.38	1.65	25
4	4		0	0	0.34	0.02	32.20	4.02	29	0.49	26.18	1.29	32
5	5		0	0	0.36	0.06	30.19	2.36	51	0.38	22.37	1.64	32
6	6		0	0	0.61	0.02	33.98	3.58	26	0.56	26.98	1.23	33
	MFCC0	MFCC1	MFCC2	MFCC3	MFCC4	MFCC5	MFCC6	MFCC7	MFCC8	MFCC9	MFCC10	MFCC11	
1	-0.60	8.90	-1.58	3.69	3.32	1.81	2.48	1.12	0.92	-1.17	-1.50	-2.00	
2	-1.31	0.28	14.18	5.34	10.28	5.39	7.45	3.48	4.48	3.93	2.47	1.17	
3	-1.46	2.82	17.38	7.79	7.49	6.53	4.07	6.27	5.35	4.94	1.98	2.57	
4	-0.41	12.20	-2.43	3.14	3.31	0.78	0.51	1.39	-0.89	0.35	-0.59	-1.45	
5	-1.79	9.83	4.12	9.38	5.34	6.55	4.36	5.32	3.50	3.83	3.43	2.45	
6	-0.89	8.43	1.50	1.38	1.09	1.41	-2.23	1.00	-2.76	-2.13	-4.75	-3.58	
	MFCC12	PERMUTATION	PPE	SHANNON	ZCR	energyentropy	spectralcentroid						
1	-3.62		2.00	0.57	12.19	0.02	3.30				0.12		
2	-0.35		2.48	0.51	12.22	0.01	3.21				0.13		
3	0.56		2.37	0.57	12.19	0.01	3.25				0.11		
4	-0.88		1.29	0.55	12.16	0.02	3.30				0.12		
5	2.25		1.58	0.55	12.22	0.01	3.24				0.10		
6	-5.13		1.31	0.53	12.13	0.02	3.30				0.13		
	spectralspread	spectralentropy	spectralrolloff	RPDE	rep								
1		0.18		0.11		0.02	0.29	1					
2		0.22		0.07		0.02	0.54	2					
3		0.21		0.02		0.01	0.35	3					
4		0.19		0.13		0.03	0.29	1					
5		0.18		0.04		0.01	0.42	2					
6		0.20		0.12		0.03	0.32	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

```
## 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(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  $w_i=1/n$ ,  $i=1,\dots,n$ 
w1 = rep(1/n,n)

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

## REPLICATION  $j=1$ :
## Algo3: Fit a classifier  $T(x_j,z)$  to the training data using weights  $w_i$ 
mod1 <- randomForest(
  x = train1 ,
  y = Yc ,
  weights = w1,
  ntree = 500,
  xtest = test1)
## summary(mod1)
```

```

## Predictions
pred1.vgam <- mod1$votes
pred1 <- apply(pred1.vgam,1,which.max)

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

```

```

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. :0.5655	Min. :0.5476	Min. :0.5476
## 1st Qu.:	:0.5536	1st Qu.:0.6190	1st Qu.:0.6131	1st Qu.:0.6190
## Median	:0.5774	Median :0.6369	Median :0.6310	Median :0.6369
## Mean	:0.5792	Mean :0.6360	Mean :0.6317	Mean :0.6382
## 3rd Qu.:	:0.6027	3rd Qu.:0.6607	3rd Qu.:0.6548	3rd Qu.:0.6548
## Max.	:0.6488	Max. :0.6964	Max. :0.7024	Max. :0.7202

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

```
## [1] 0.03285949 0.02841115 0.03181675 0.03108869
```

```
summary(TestAccuracy)
```

##	V1	V2	V3	V4
## Min.	:0.4737	Min. :0.4912	Min. :0.4912	Min. :0.5088
## 1st Qu.:	:0.5439	1st Qu.:0.6140	1st Qu.:0.5965	1st Qu.:0.5965
## Median	:0.5965	Median :0.6491	Median :0.6316	Median :0.6404
## Mean	:0.5954	Mean :0.6458	Mean :0.6358	Mean :0.6433
## 3rd Qu.:	:0.6316	3rd Qu.:0.6842	3rd Qu.:0.6842	3rd Qu.:0.6886
## Max.	:0.7368	Max. :0.7719	Max. :0.7895	Max. :0.7719

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

```
## [1] 0.06112160 0.05590811 0.05429586 0.06342977
```

```
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.	:0.5009	Min. :0.5640	Min. :0.5486	Min. :0.5471
## 1st Qu.:	:0.5529	1st Qu.:0.6196	1st Qu.:0.6151	1st Qu.:0.6220
## Median	:0.5756	Median :0.6422	Median :0.6368	Median :0.6415
## Mean	:0.5779	Mean :0.6394	Mean :0.6363	Mean :0.6411
## 3rd Qu.:	:0.6034	3rd Qu.:0.6603	3rd Qu.:0.6616	3rd Qu.:0.6570
## Max.	:0.6479	Max. :0.6996	Max. :0.7121	Max. :0.7280

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

```
## [1] 0.03371021 0.02820254 0.03296666 0.03140664
```

```
summary(TestPrecisionMacroAve)
```

##	V1	V2	V3	V4
## Min.	:0.4548	Min. :0.4690	Min. :0.5144	Min. :0.4980
## 1st Qu.:	:0.5659	1st Qu.:0.6214	1st Qu.:0.6104	1st Qu.:0.6082
## Median	:0.6000	Median :0.6638	Median :0.6414	Median :0.6484
## Mean	:0.6018	Mean :0.6569	Mean :0.6455	Mean :0.6533
## 3rd Qu.:	:0.6419	3rd Qu.:0.6933	3rd Qu.:0.6900	3rd Qu.:0.7056
## Max.	:0.7392	Max. :0.7869	Max. :0.8005	Max. :0.7828

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

```
## [1] 0.06434599 0.05714055 0.05564299 0.06507268
```

```
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. :0.5655	Min. :0.5476	Min. :0.5476
## 1st Qu.	:0.5536	1st Qu.:0.6190	1st Qu.:0.6131	1st Qu.:0.6190
## Median	:0.5774	Median :0.6369	Median :0.6310	Median :0.6369
## Mean	:0.5792	Mean :0.6360	Mean :0.6317	Mean :0.6382
## 3rd Qu.	:0.6027	3rd Qu.:0.6607	3rd Qu.:0.6548	3rd Qu.:0.6548
## Max.	:0.6488	Max. :0.6964	Max. :0.7024	Max. :0.7202

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

```
## [1] 0.03285949 0.02841115 0.03181675 0.03108869
```

```
summary(TestRecallMacroAve)
```

##	V1	V2	V3	V4
## Min.	:0.4737	Min. :0.4912	Min. :0.4912	Min. :0.5088
## 1st Qu.	:0.5439	1st Qu.:0.6140	1st Qu.:0.5965	1st Qu.:0.5965
## Median	:0.5965	Median :0.6491	Median :0.6316	Median :0.6404
## Mean	:0.5954	Mean :0.6458	Mean :0.6358	Mean :0.6433
## 3rd Qu.	:0.6316	3rd Qu.:0.6842	3rd Qu.:0.6842	3rd Qu.:0.6886
## Max.	:0.7368	Max. :0.7719	Max. :0.7895	Max. :0.7719

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

```
## [1] 0.06112160 0.05590811 0.05429586 0.06342977
```

```
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. :0.5018	Min. :0.5644	Min. :0.5473	Min. :0.5472
##	1st Qu.:0.5536	1st Qu.:0.6181	1st Qu.:0.6131	1st Qu.:0.6194
##	Median :0.5764	Median :0.6387	Median :0.6320	Median :0.6382
##	Mean :0.5778	Mean :0.6366	Mean :0.6324	Mean :0.6388
##	3rd Qu.:0.6027	3rd Qu.:0.6586	3rd Qu.:0.6553	3rd Qu.:0.6548
##	Max. :0.6479	Max. :0.6975	Max. :0.7041	Max. :0.7217

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

```
## [1] 0.03333531 0.02825698 0.03219113 0.03108633
```

```
summary(TestF1ScoreMacroAve)
```

##	V1	V2	V3	V4
##	Min. :0.4598	Min. :0.4698	Min. :0.4948	Min. :0.4976
##	1st Qu.:0.5494	1st Qu.:0.6100	1st Qu.:0.5971	1st Qu.:0.5999
##	Median :0.5944	Median :0.6492	Median :0.6318	Median :0.6393
##	Mean :0.5923	Mean :0.6451	Mean :0.6339	Mean :0.6424
##	3rd Qu.:0.6329	3rd Qu.:0.6827	3rd Qu.:0.6758	3rd Qu.:0.6903
##	Max. :0.7377	Max. :0.7740	Max. :0.7910	Max. :0.7729

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

```
## [1] 0.06197273 0.05612877 0.05430764 0.06405975
```

```
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"))
```