

Support Vector Machine (SVM) with Linear kernel

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(e1071)
## 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_SVM_poly3_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_SVM_poly3_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_SVM_poly3_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. : 0.00
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
GNP	HNH	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

```
## 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 $w_i=1/n$, $i=1,...,n$
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 <- tune( "svm", STATUS_fact ~ . ,
              data = train1,
              weights = w1,
              kernel = "polynomial", degree=3,
              ranges = list(cost=c(0.01,0.1,0.5,1,5,10,20,50)) )
## summary(mod1)
mejor_mod1 <- mod1$best.model
## Predictions
```

```

pred1 <- predict(mejor_mod1, newdata = train1)

## 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 <- tune( "svm", STATUS_fact ~ . ,
              data = train2,
              weights = wi2,
              kernel = "polynomial", degree=3,
              ranges = list(cost=c(0.01,0.1,0.5,1,5,10,20,50)) )
## summary(mod2)
mejor_mod2 <- mod2$best.model
## Predictions
pred2 <- predict(mejor_mod2, newdata = train2)

## 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 <- tune( "svm", STATUS_fact ~ . ,
              data = train3,
              weights = wi3,
              kernel = "polynomial", degree=3,
              ranges = list(cost=c(0.01,0.1,0.5,1,5,10,20,50)) )
## summary(mod3)
mejor_mod3 <- mod3$best.model
## Predictions
pred3 <- predict(mejor_mod3, newdata = train3)

## Algo4: Compute $err = \sum wi I[Y \neq T(xj,z)] / \sum wi$
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  $w_i = w_i \cdot \exp(\alpha \cdot I[Y \neq T(x_j, z)])$ 
wi4 = wi3*exp(alp3*(Yc!=pred3))
## Algo7: Re-normalize  $w_i$ 
wi4 = c(wi4/sum(wi4))

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

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

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 <- predict(mejor_mod1, newdata = test1)
pred2.new <- predict(mejor_mod2, newdata = test2)
pred3.new <- predict(mejor_mod3, newdata = test3)

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.	:0.7024	Min. :0.5893	Min. :0.7679	Min. :0.7738
## 1st Qu.:	0.9405	1st Qu.:0.9524	1st Qu.:0.9524	1st Qu.:0.9628
## Median	:0.9554	Median :0.9762	Median :0.9881	Median :0.9851
## Mean	:0.9493	Mean :0.9567	Mean :0.9678	Mean :0.9760
## 3rd Qu.:	0.9881	3rd Qu.:0.9940	3rd Qu.:1.0000	3rd Qu.:0.9940
## Max.	:1.0000	Max. :1.0000	Max. :1.0000	Max. :1.0000

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

```
## [1] 0.05892848 0.06777183 0.04771736 0.02818603
```

```
summary(TestAccuracy)
```

##	V1	V2	V3	V4
## Min.	:0.4386	Min. :0.3860	Min. :0.4386	Min. :0.4386
## 1st Qu.:	0.5263	1st Qu.:0.4912	1st Qu.:0.5439	1st Qu.:0.5439
## Median	:0.5614	Median :0.5263	Median :0.5789	Median :0.5877
## Mean	:0.5619	Mean :0.5233	Mean :0.5837	Mean :0.5828
## 3rd Qu.:	0.5965	3rd Qu.:0.5614	3rd Qu.:0.6184	3rd Qu.:0.6140
## Max.	:0.6842	Max. :0.6491	Max. :0.7193	Max. :0.7368

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

```
## [1] 0.04863783 0.05429213 0.06193186 0.05903932
```

```
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.	:0.8157	Min. :0.8053	Min. :0.8503	Min. :0.8503
## 1st Qu.:	0.9471	1st Qu.:0.9579	1st Qu.:0.9564	1st Qu.:0.9654
## Median	:0.9598	Median :0.9774	Median :0.9881	Median :0.9856
## Mean	:0.9592	Mean :0.9661	Mean :0.9734	Mean :0.9785
## 3rd Qu.:	0.9885	3rd Qu.:0.9942	3rd Qu.:1.0000	3rd Qu.:0.9942
## Max.	:1.0000	Max. :1.0000	Max. :1.0000	Max. :1.0000

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

```
## [1] 0.03772978 0.04156846 0.03305215 0.02129204
```

```
summary(TestPrecisionMacroAve)
```

##	V1	V2	V3	V4
## Min.	:0.4514	Min. :0.4015	Min. :0.4345	Min. :0.4383
## 1st Qu.:	0.5560	1st Qu.:0.5027	1st Qu.:0.5574	1st Qu.:0.5575
## Median	:0.6018	Median :0.5414	Median :0.5965	Median :0.6068
## Mean	:0.5985	Mean :0.5416	Mean :0.6001	Mean :0.6019
## 3rd Qu.:	0.6419	3rd Qu.:0.5864	3rd Qu.:0.6476	3rd Qu.:0.6478
## Max.	:0.7551	Max. :0.7311	Max. :0.7452	Max. :0.7431

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

```
## [1] 0.06407046 0.06507355 0.06715995 0.06498484
```

```
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.	:0.7024	Min. :0.5893	Min. :0.7679	Min. :0.7738
## 1st Qu.:	0.9405	1st Qu.:0.9524	1st Qu.:0.9524	1st Qu.:0.9628
## Median	:0.9554	Median :0.9762	Median :0.9881	Median :0.9851
## Mean	:0.9493	Mean :0.9567	Mean :0.9678	Mean :0.9760
## 3rd Qu.:	0.9881	3rd Qu.:0.9940	3rd Qu.:1.0000	3rd Qu.:0.9940
## Max.	:1.0000	Max. :1.0000	Max. :1.0000	Max. :1.0000

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

```
## [1] 0.05892848 0.06777183 0.04771736 0.02818603
```

```
summary(TestRecallMacroAve)
```

##	V1	V2	V3	V4
## Min.	:0.4386	Min. :0.3860	Min. :0.4386	Min. :0.4386
## 1st Qu.:	0.5263	1st Qu.:0.4912	1st Qu.:0.5439	1st Qu.:0.5439
## Median	:0.5614	Median :0.5263	Median :0.5789	Median :0.5877
## Mean	:0.5619	Mean :0.5233	Mean :0.5837	Mean :0.5828
## 3rd Qu.:	0.5965	3rd Qu.:0.5614	3rd Qu.:0.6184	3rd Qu.:0.6140
## Max.	:0.6842	Max. :0.6491	Max. :0.7193	Max. :0.7368

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

```
## [1] 0.04863783 0.05429213 0.06193186 0.05903932
```

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

F1-Score Macro Average

```
summary(FitF1ScoreMacroAve)
```

##	V1	V2	V3	V4
##	Min. :0.6992	Min. :0.5653	Min. :0.7691	Min. :0.7747
##	1st Qu.:0.9408	1st Qu.:0.9527	1st Qu.:0.9528	1st Qu.:0.9629
##	Median :0.9556	Median :0.9763	Median :0.9881	Median :0.9851
##	Mean :0.9496	Mean :0.9569	Mean :0.9681	Mean :0.9761
##	3rd Qu.:0.9881	3rd Qu.:0.9940	3rd Qu.:1.0000	3rd Qu.:0.9940
##	Max. :1.0000	Max. :1.0000	Max. :1.0000	Max. :1.0000

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

```
## [1] 0.05893278 0.06836732 0.04717544 0.02801302
```

```
summary(TestF1ScoreMacroAve)
```

##	V1	V2	V3	V4
##	Min. :0.3989	Min. :0.3954	Min. :0.4280	Min. :0.4358
##	1st Qu.:0.5143	1st Qu.:0.4787	1st Qu.:0.5411	1st Qu.:0.5370
##	Median :0.5538	Median :0.5160	Median :0.5771	Median :0.5816
##	Mean :0.5459	Mean :0.5179	Mean :0.5794	Mean :0.5773
##	3rd Qu.:0.5886	3rd Qu.:0.5600	3rd Qu.:0.6166	3rd Qu.:0.6094
##	Max. :0.6781	Max. :0.6485	Max. :0.7207	Max. :0.7374

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

```
## [1] 0.05543536 0.05528654 0.06401131 0.06098112
```

```
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"))
```