

# Multinomial linear regression (MLR)

Feature space partition ensemble model for replication (FESPAE)

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(VGAM)
## 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 = "FESPAE_crossval_strata_allvar_MLR_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 = "FESPAE_crossval_strata_allvar_MLR_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 = "FESPAE_crossval_strata_allvar_MLR_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),
senenergyentropy= 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,
senenergyentropy, sspectralcentroid, sspectralspread,
sspectralentropy, sspectralrolloff, sRPDE,
STATUS_fact,SEX, rep,ID_fact)
```

# Crossvalidation

## Subspaces

```
## Partition of subspaces
## The feature space is randomly partitioned into K subspaces with roughly equal sizes
## k = number of predictors
## K = subspaces

K0 = 4 ## sub-spaces
k = 32 ## explanatory variables
k2 = round(k/K0)
space = 1:k
subspaces = rep(list(rep(NA,k2)),K0) ## Subspaces
set.seed(12345)
for(j in 1:(K0-1)){
  space1 = sample(space, size=k2, replace=FALSE)
  space = setdiff(space,space1)
  subspaces[[j]] = space1[order(space1)]
}
space1 = space
subspaces[[K0]] = space1[order(space1)]
## 32 features = 1x32, 2x16, 4x8,
subspaces
```

```
## [[1]]
## [1] 11 14 16 19 24 26 28 29
##
## [[2]]
## [1] 2 6 7 10 12 21 30 32
##
## [[3]]
## [1] 1 4 5 9 13 15 27 31
##
## [[4]]
## [1] 3 8 17 18 20 22 23 25
```

## 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,1)) ## Accuracy Rate
FitPrecisionClass = TestPrecisionClass <- array(NA,dim=c(SIM,1,3)) ## Precision per class
FitRecallClass = TestRecallClass <- array(NA,dim=c(SIM,1,3)) ## Recall per class
FitF1ScoreClass = TestF1ScoreClass <- array(NA,dim=c(SIM,1,3)) ## F1-score per class
FitPrecisionMacroAve = TestPrecisionMacroAve <- array(NA,dim=c(SIM,1)) ## Precision Macro Average
FitRecallMacroAve = TestRecallMacroAve <- array(NA,dim=c(SIM,1)) ## Recall Macro Average
FitF1ScoreMacroAve = TestF1ScoreMacroAve <- array(NA,dim=c(SIM,1)) ## 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))  
test2 <- test2 %>% select(-c(rep,ID_fact))  
test3 <- test3 %>% select(-c(rep,ID_fact))  
  
##-----  
## Algorithm FESPAE  
## Feature space partition ensemble model for replication  
##-----  
  
## Algo1: The feature space is randomly partitioned into M subspaces, {S1,S2,...,SM}  
  
K0 = 4 ## sub-spaces  
k = 32 ## explanatory variables  
k2 = round(k/K0)  
space = 1:k  
subspaces = rep(list(rep(NA,k2)),K0) ## Subspaces  
set.seed(12345)  
for(j in 1:(K0-1)){  
  space1 = sample(space, size=k2, replace=FALSE)  
  space = setdiff(space,space1)  
  subspaces[[j]] = space1[order(space1)]  
}  
space1 = space
```



```

subspaces[[K0]] = space1[order(space1)]
# 32 features = 1x32, 2x16, 4x8,

##-----
## Algo2: for feature subspace m = 1 to M do

pred.vgam = array(NA,dim=c(n,G,K0,3)) ### 3 repeticiones
pred.new.vgam = array(NA,dim=c(n.new,G,K0,3)) ### 3 repeticiones
##-----
## Algo3: for replication j = 1 to J do

## REPLICATION j=1:
for(parti1 in 1:K0){ ## partition of the subspaces
  train1_par = train1[,c(subspaces[[parti1]],k+1)]
  test1_par = test1[,c(subspaces[[parti1]],k+1)]

## Algo4: Fit a classifier  $T(x_j, z)$ ,  $x_j \in S_m$ , to the training data
  mod1 <- vglm( STATUS_fact ~ . ,
               multinomial(refLevel=1),
               data = train1_par)
## summary(mod1)

## Algo5: Compute the C response probabilities  $\{\pi^{(m,j)}_{ic}\}$ , for  $i=1, \dots, n$ .
### Predictions
  predict1.vgam <- predict(mod1, newdata = train1_par, "response")
  predict1.new.vgam <- predict(mod1, newdata = test1_par, "response")

  pred.vgam[, , parti1, 1] = predict1.vgam
  pred.new.vgam[, , parti1, 1] = predict1.new.vgam
}

## REPLICATION j=2:
for(parti2 in 1:K0){ ## partition of the subspaces
  train2_par = train2[,c(subspaces[[parti2]],k+1)]
  test2_par = test2[,c(subspaces[[parti2]],k+1)]

## Algo4: Fit a classifier  $T(x_j, z)$ ,  $x_j \in S_m$ , to the training data
  mod2 <- vglm( STATUS_fact ~ . ,
               multinomial(refLevel=1),
               data = train2_par)
## summary(mod2)

## Algo5: Compute the C response probabilities  $\{\pi^{(m,j)}_{ic}\}$ , for  $i=1, \dots, n$ .
### Predictions
  predict2.vgam <- predict(mod2, newdata = train2_par, "response")
  predict2.new.vgam <- predict(mod2, newdata = test2_par, "response")

  pred.vgam[, , parti2, 2] = predict2.vgam
  pred.new.vgam[, , parti2, 2] = predict2.new.vgam
}

```

```

## REPLICATION j=3:
for(parti3 in 1:K0){ ## partition of the subspaces
  train3_par = train3[,c(subspaces[[parti3]],k+1)]
  test3_par = test3[,c(subspaces[[parti3]],k+1)]

## Algo4: Fit a classifier  $T(x_j, z)$ ,  $x_j \in S_m$ , to the training data
  mod3 <- vglm( STATUS_fact ~ . ,
               multinomial(refLevel=1),
               data = train3_par)
## summary(mod3)

## Algo5: Compute the C response probabilities  $\{\pi^{(m,j)}_{ic}\}$ , for  $i=1, \dots, n$ .
### Predictions
predict3.vgam <- predict(mod3, newdata = train3_par, "response")
predict3.new.vgam <- predict(mod3, newdata = test3_par, "response")

pred.vgam[, , parti3, 3] = predict3.vgam
pred.new.vgam[, , parti3, 3] = predict3.new.vgam
}

##-----
## Algo6: End for replication  $j = 1$  to  $J$ 
## Algo7: End for feature subspace  $m = 1$  to  $M$ 
##-----
## Algo8: Output: compute the response probabilities  $\pi_{ic} = \text{mean}(\{\pi^{(m,j)}_{ic}\})$ 
pred.ave.vgam = apply(pred.vgam, c(1,2), mean)

### Predict new subjects
pred.ave.new.vgam = apply(pred.new.vgam, c(1,2), mean)

##-----
## Algo8: Output: compute the response category  $T^*(x, z) = \arg \max \{\pi_{ic}\}$ 
pred.vgam_max <- apply(pred.ave.vgam, 1, which.max)

### Predict new subjects
pred.new.vgam_max <- apply(pred.ave.new.vgam, 1, which.max)

##-----
## End FESPAE
##-----
## Classification Metrics for models predicting nominal outcomes

## Accuracy Rate
FitAccuracy[sim,] = c(sum(Yc==pred.vgam_max)/n)

TestAccuracy[sim,] = c(sum(Yc.new==pred.new.vgam_max)/n.new)

## Precision
for(cate in 1:3){
  FitPrecisionClass[sim,1, cate] = fn_precision_class(Yc, pred.vgam_max, cate)
  TestPrecisionClass[sim,1, cate] = fn_precision_class(Yc.new, pred.new.vgam_max, cate)
}

```

```

}
FitPrecisionMacroAve[sim, 1] = mean(FitPrecisionClass[sim, 1,])
TestPrecisionMacroAve[sim,1] = mean(TestPrecisionClass[sim,1,])

## Recall
for(cate in 1:3){
  FitRecallClass[sim,1, cate] = fn_recall_class(Yc, pred.vgam_max, cate)
  TestRecallClass[sim,1, cate] = fn_recall_class(Yc.new, pred.new.vgam_max, cate)
}
FitRecallMacroAve[sim, 1] = mean(FitRecallClass[sim, 1,])
TestRecallMacroAve[sim,1] = mean(TestRecallClass[sim,1,])

## F1-Score
for(cate in 1:3){
  FitF1ScoreClass[sim,1, cate]= fn_f1score_class(Yc, pred.vgam_max, cate)
  TestF1ScoreClass[sim,1, cate] = fn_f1score_class(Yc.new, pred.new.vgam_max, cate)
}
FitF1ScoreMacroAve[sim, 1] = mean(FitF1ScoreClass[sim, 1,])
TestF1ScoreMacroAve[sim,1] = mean(TestF1ScoreClass[sim,1,])

##-----
} ## END sim
##-----

```

# Results

## Accuracy Rate

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

summary(FitAccuracy)
```

```
##           V1
##  Min.      :0.6845
## 1st Qu.:0.7202
##  Median :0.7321
##   Mean   :0.7341
## 3rd Qu.:0.7455
##   Max.   :0.7917
```

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

```
## [1] 0.02046381
```

```
summary(TestAccuracy)
```

```
##           V1
##  Min.      :0.4912
## 1st Qu.:0.5965
##  Median :0.6316
##   Mean   :0.6333
## 3rd Qu.:0.6667
##   Max.   :0.7193
```

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

```
## [1] 0.05070611
```

```
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
##  Min.    :0.6869
## 1st Qu.:0.7197
##  Median :0.7330
##   Mean  :0.7341
## 3rd Qu.:0.7470
##   Max.  :0.7937
```

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

```
## [1] 0.02055901
```

```
summary(TestPrecisionMacroAve)
```

```
##          V1
##  Min.    :0.4962
## 1st Qu.:0.6024
##  Median :0.6377
##   Mean  :0.6383
## 3rd Qu.:0.6808
##   Max.  :0.7417
```

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

```
## [1] 0.05261503
```

```
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
##  Min.    :0.6845
## 1st Qu.:0.7202
##  Median :0.7321
##   Mean   :0.7341
## 3rd Qu.:0.7455
##   Max.   :0.7917
```

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

```
## [1] 0.02046381
```

```
summary(TestRecallMacroAve)
```

```
##          V1
##  Min.    :0.4912
## 1st Qu.:0.5965
##  Median :0.6316
##   Mean   :0.6333
## 3rd Qu.:0.6667
##   Max.   :0.7193
```

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

```
## [1] 0.05070611
```

```
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
##  Min.    :0.6850
## 1st Qu.:0.7195
##  Median :0.7321
##   Mean   :0.7334
## 3rd Qu.:0.7453
##   Max.   :0.7916
```

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

```
## [1] 0.02057917
```

```
summary(TestF1ScoreMacroAve)
```

```
##          V1
##  Min.    :0.4919
## 1st Qu.:0.5956
##  Median :0.6307
##   Mean   :0.6306
## 3rd Qu.:0.6692
##   Max.   :0.7233
```

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

```
## [1] 0.05159972
```

```
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"))
```