Support Vector Machine (SVM) with Polynomial kernel

Replication-based stagewise additive modeling (RSAM)

Simulation-based settings

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

Data

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

archivo = "RSAM_crossval_strata_SVM_poly3_simula"   ## name of the files to save results
```

```
dim(datos2)
```

```
[1] 900  24
```

```
summary(datos2)
```

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

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

```
head(datos2)
```

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

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

```
  1   2   3
```

```
300 300 300
```

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

Crossvalidation

Training and testing data subsets

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

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

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

Classification metrics for models predicting nominal outcomes

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

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

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

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

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

Model estimation

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

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

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

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

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

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

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

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

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

## REPLICATION j=1:
## Algo3: Fit a classifier $T(xj,z)$ to the training data using weights $wi$
mod1 <- tune( "svm",  STATUS_fact ~ . ,
              data = train1,
              weights = wi1,
              kernel = "polynomial", degree=3,
              ranges = list(cost=c(0.001,0.01,0.1,0.5,1,25,10,20,50)) )
## summary(mod1)
mejor_mod1 <- mod1$best.model
## Predictions
```

```
pred1 <- predict(mejor_mod1, newdata = train1)

## 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 <- tune( "svm",  STATUS_fact ~ . ,
              data = train2,
              weights = wi2,
              kernel = "polynomial", degree=3,
              ranges = list(cost=c(0.001,0.01,0.1,0.5,1,25,10,20,50)) )
## summary(mod2)
mejor_mod2 <- mod2$best.model
## Predictions
pred2 <- predict(mejor_mod2, newdata = train2)

## 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 <- tune( "svm",  STATUS_fact ~ . ,
              data = train3,
              weights = wi3,
              kernel = "polynomial", degree=3,
              ranges = list(cost=c(0.001,0.01,0.1,0.5,1,2,5,10,20,50)) )
## summary(mod3)
mejor_mod3 <- mod3$best.model
## Predictions
pred3 <- predict(mejor_mod3, newdata = train3)

## Algo4: Compute $err = \sum wi I[Y != T(xj,z)] / \sum wi$
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 <- 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.   :0.9733   Min.   :0.9378   Min.   :0.9467   Min.   :0.9822
##  1st Qu.:0.9867   1st Qu.:0.9867   1st Qu.:0.9778   1st Qu.:1.0000
##  Median :0.9867   Median :0.9867   Median :0.9822   Median :1.0000
##  Mean   :0.9882   Mean   :0.9876   Mean   :0.9835   Mean   :0.9990
##  3rd Qu.:0.9911   3rd Qu.:0.9911   3rd Qu.:0.9867   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

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

```
## [1] 0.005831572 0.008394322 0.007556216 0.003923702
```

```
summary(TestAccuracy)
```

```
##        V1               V2               V3               V4
##  Min.   :0.6933   Min.   :0.7867   Min.   :0.7600   Min.   :0.7867
##  1st Qu.:0.7567   1st Qu.:0.8133   1st Qu.:0.8133   1st Qu.:0.9067
##  Median :0.7733   Median :0.8400   Median :0.8400   Median :0.9200
##  Mean   :0.7781   Mean   :0.8396   Mean   :0.8389   Mean   :0.9175
##  3rd Qu.:0.8000   3rd Qu.:0.8533   3rd Qu.:0.8667   3rd Qu.:0.9333
##  Max.   :0.8800   Max.   :0.9467   Max.   :0.9200   Max.   :0.9733
```

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

```
## [1] 0.03754629 0.03081854 0.03579114 0.03106461
```

```
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.   :0.9734   Min.   :0.9379   Min.   :0.9475   Min.   :0.9825
##  1st Qu.:0.9867   1st Qu.:0.9867   1st Qu.:0.9792   1st Qu.:1.0000
##  Median :0.9872   Median :0.9872   Median :0.9831   Median :1.0000
##  Mean   :0.9883   Mean   :0.9878   Mean   :0.9839   Mean   :0.9991
##  3rd Qu.:0.9912   3rd Qu.:0.9913   3rd Qu.:0.9872   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

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

```
## [1] 0.005768325 0.008332723 0.007335743 0.003788969
```

```
summary(TestPrecisionMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.7044   Min.   :0.7846   Min.   :0.7611   Min.   :0.7870
##  1st Qu.:0.7592   1st Qu.:0.8210   1st Qu.:0.8144   1st Qu.:0.9070
##  Median :0.7868   Median :0.8435   Median :0.8433   Median :0.9228
##  Mean   :0.7836   Mean   :0.8452   Mean   :0.8425   Mean   :0.9199
##  3rd Qu.:0.8040   3rd Qu.:0.8645   3rd Qu.:0.8689   3rd Qu.:0.9354
##  Max.   :0.8800   Max.   :0.9491   Max.   :0.9245   Max.   :0.9753
```

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

```
## [1] 0.03819301 0.03025447 0.03607625 0.03040902
```

```
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.   :0.9733   Min.   :0.9378   Min.   :0.9467   Min.   :0.9822
##  1st Qu.:0.9867   1st Qu.:0.9867   1st Qu.:0.9778   1st Qu.:1.0000
##  Median :0.9867   Median :0.9867   Median :0.9822   Median :1.0000
##  Mean   :0.9882   Mean   :0.9876   Mean   :0.9835   Mean   :0.9990
##  3rd Qu.:0.9911   3rd Qu.:0.9911   3rd Qu.:0.9867   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

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

```
## [1] 0.005831572 0.008394322 0.007556216 0.003923702
```

```
summary(TestRecallMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.6933   Min.   :0.7867   Min.   :0.7600   Min.   :0.7867
##  1st Qu.:0.7567   1st Qu.:0.8133   1st Qu.:0.8133   1st Qu.:0.9067
##  Median :0.7733   Median :0.8400   Median :0.8400   Median :0.9200
##  Mean   :0.7781   Mean   :0.8396   Mean   :0.8389   Mean   :0.9175
##  3rd Qu.:0.8000   3rd Qu.:0.8533   3rd Qu.:0.8667   3rd Qu.:0.9333
##  Max.   :0.8800   Max.   :0.9467   Max.   :0.9200   Max.   :0.9733
```

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

```
## [1] 0.03754629 0.03081854 0.03579114 0.03106461
```

```
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.   :0.9733   Min.   :0.9377   Min.   :0.9466   Min.   :0.9822
##  1st Qu.:0.9866   1st Qu.:0.9866   1st Qu.:0.9779   1st Qu.:1.0000
##  Median :0.9867   Median :0.9867   Median :0.9823   Median :1.0000
##  Mean   :0.9882   Mean   :0.9877   Mean   :0.9835   Mean   :0.9990
##  3rd Qu.:0.9911   3rd Qu.:0.9911   3rd Qu.:0.9867   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

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

```
## [1] 0.005824462 0.008391914 0.007536770 0.003912297
```

```
summary(TestF1ScoreMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.6940   Min.   :0.7848   Min.   :0.7594   Min.   :0.7862
##  1st Qu.:0.7525   1st Qu.:0.8143   1st Qu.:0.8125   1st Qu.:0.9056
##  Median :0.7738   Median :0.8397   Median :0.8399   Median :0.9200
##  Mean   :0.7773   Mean   :0.8395   Mean   :0.8385   Mean   :0.9173
##  3rd Qu.:0.8005   3rd Qu.:0.8556   3rd Qu.:0.8657   3rd Qu.:0.9335
##  Max.   :0.8800   Max.   :0.9471   Max.   :0.9204   Max.   :0.9736
```

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

```
## [1] 0.03781575 0.03098335 0.03598943 0.03110443
```

```
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"))
```