Support Vector Machine (SVM) Plynomial kernel

Feature space partition ensemble model for replication (FESPAE)

Simulation-based settings

Data

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

archivo = "FESPAE_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

Subspaces

```r
## Function to compute the Mode
Mode <- function(x, na.rm = FALSE) {
  if(na.rm){
    x = x[!is.na(x)]
    }
  ux <- unique(x)
  return(ux[which.max(tabulate(match(x, ux)))])
}


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

K0 = 3 ### sub-spaces
k = 21 ### explanatory variable
k2 = round(k/K0)
space = 1:k
subspaces = rep(list(rep(NA,k2)),K0)
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)]
# 21 features = 1x21, 3x7,
subspaces
```

```
## [[1]]
## [1]   2   6 11 14 16 18 19
##
## [[2]]
## [1]   1   8   9 10 13 20 21
##
## [[3]]
## [1]   3   4   5   7 12 15 17
```

Training and testing data subsets

```r
## 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

```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(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 = 3   ## sub-spaces
k = 21   ## 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
```

```r
subspaces[[K0]] = space1[order(space1)]
# 21 features = 1x21, 3x7,


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

pred.vgam = array(NA,dim=c(n,K0,3))  ## 3 repetitions
pred.new.vgam = array(NA,dim=c(n.new,K0,3))  ## 3 repetitions
##-----------------------------------------------------
## 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 $T(xj,z)$, $xj\in Sm$, to the training data
mod1 <- tune( "svm",  STATUS_fact ~ . ,
              data = train1_par,
              kernel = "polynomial", degree=3,
              ranges = list(cost=c(0.001,0.01,0.1,0.5,1,2,5,10,20,50)) )
## summary(mod1)
mejor_mod1 <- mod1$best.model

## Algo5: Compute the C response probabilities {\pi^(m,j)_{ic}}, for i=1,...,n.
## Predictions
predict1 <- predict(mejor_mod1, newdata = train1_par)
predict1.new <- predict(mejor_mod1, newdata = test1_par)

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


## 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 $T(xj,z)$, $xj\in Sm$, to the training data
mod2 <- tune( "svm",  STATUS_fact ~ . ,
              data = train2_par,
              kernel = "polynomial", degree=3,
              ranges = list(cost=c(0.001,0.01,0.1,0.5,1,2,5,10,20,50)) )
## summary(mod2)
mejor_mod2 <- mod2$best.model

## Algo5: Compute the C response probabilities {\pi^(m,j)_{ic}}, for i=1,...,n.
## Predictions
predict2 <- predict(mejor_mod2, newdata = train2_par)
predict2.new <- predict(mejor_mod2, newdata = test2_par)

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

```r
}

## 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 $T(xj,z)$, $xj\in Sm$, to the training data
mod3 <- tune( "svm",  STATUS_fact ~ . ,
             data = train3_par,
             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

## Algo5: Compute the C response probabilities {\pi^(m,j)_{ic}}, for i=1,...,n.
## Predictions
predict3 <- predict(mejor_mod3, newdata = train3_par)
predict3.new <- predict(mejor_mod3, newdata = test3_par)

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


##--------------------------------------------------
## 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} = mean({\pi^(m,j)_{ic}})
##--------------------------------------------------
## Algo8: Output: compute the response category T*(x,z) = arg max {\pi_{ic}}

pred.max = array(NA,dim=c(n,3))
pred.vgam_max = array(NA,dim=n)
for(i in 1:n){
  for(j in 1:3){
    pred.max[i,j] = Mode(pred.vgam[i,,j])
  }
  pred.vgam_max[i] = Mode(pred.vgam[i,,])
}
### Predict new subjects
pred.new.max = array(NA,dim=c(n.new,3))
pred.new.vgam_max = array(NA,dim=n.new)
for(i in 1:n.new){
  for(j in 1:3){
    pred.new.max[i,j] = Mode(pred.new.vgam[i,,j])
  }
  pred.new.vgam_max[i] = Mode(pred.new.vgam[i,,])
}


##--------------------------------------------------
## End FESPAE
##--------------------------------------------------
```

```r
## Classification Metrics for models predicting nominal outcomes


## Accuracy Rate
FitAccuracy[sim,]= c(sum(Yc==pred.max[,1])/n,
                     sum(Yc==pred.max[,2])/n,
                     sum(Yc==pred.max[,3])/n,
                     sum(Yc==pred.vgam_max)/n)
TestAccuracy[sim,] = c(sum(Yc.new==pred.new.max[,1])/n.new,
                       sum(Yc.new==pred.new.max[,2])/n.new,
                       sum(Yc.new==pred.new.max[,3])/n.new,
                       sum(Yc.new==pred.new.vgam_max)/n.new)


## Precision
for(cate in 1:3){
  FitPrecisionClass[sim,1, cate] = fn_precision_class(Yc, pred.max[,1], cate)
  FitPrecisionClass[sim,2, cate] = fn_precision_class(Yc, pred.max[,2], cate)
  FitPrecisionClass[sim,3, cate] = fn_precision_class(Yc, pred.max[,3], cate)
  FitPrecisionClass[sim,4, cate] = fn_precision_class(Yc, pred.vgam_max, cate)

  TestPrecisionClass[sim,1, cate] = fn_precision_class(Yc.new, pred.new.max[,1], cate)
  TestPrecisionClass[sim,2, cate] = fn_precision_class(Yc.new, pred.new.max[,2], cate)
  TestPrecisionClass[sim,3, cate] = fn_precision_class(Yc.new, pred.new.max[,3], cate)
  TestPrecisionClass[sim,4, cate] = fn_precision_class(Yc.new, pred.new.vgam_max, 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, pred.max[,1], cate)
  FitRecallClass[sim,2, cate] = fn_recall_class(Yc, pred.max[,2], cate)
  FitRecallClass[sim,3, cate] = fn_recall_class(Yc, pred.max[,3], cate)
  FitRecallClass[sim,4, cate] = fn_recall_class(Yc, pred.vgam_max, cate)

  TestRecallClass[sim,1, cate] = fn_recall_class(Yc.new, pred.new.max[,1], cate)
  TestRecallClass[sim,2, cate] = fn_recall_class(Yc.new, pred.new.max[,2], cate)
  TestRecallClass[sim,3, cate] = fn_recall_class(Yc.new, pred.new.max[,3], cate)
  TestRecallClass[sim,4, cate] = fn_recall_class(Yc.new, pred.new.vgam_max, 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, pred.max[,1], cate)
  FitF1ScoreClass[sim,2, cate]= fn_f1score_class(Yc, pred.max[,2], cate)
  FitF1ScoreClass[sim,3, cate]= fn_f1score_class(Yc, pred.max[,3], cate)
  FitF1ScoreClass[sim,4, cate]= fn_f1score_class(Yc, pred.vgam_max, cate)
```

```
  TestF1ScoreClass[sim,1, cate] = fn_f1score_class(Yc.new, pred.new.max[,1], cate)
  TestF1ScoreClass[sim,2, cate] = fn_f1score_class(Yc.new, pred.new.max[,2], cate)
  TestF1ScoreClass[sim,3, cate] = fn_f1score_class(Yc.new, pred.new.max[,3], cate)
  TestF1ScoreClass[sim,4, cate] = fn_f1score_class(Yc.new, pred.new.vgam_max, 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.   :0.8356   Min.   :0.8933   Min.   :0.8622   Min.   :0.9867
## 1st Qu.:0.8800   1st Qu.:0.9333   1st Qu.:0.9022   1st Qu.:0.9956
## Median :0.8978   Median :0.9511   Median :0.9222   Median :0.9956
## Mean   :0.8966   Mean   :0.9501   Mean   :0.9225   Mean   :0.9971
## 3rd Qu.:0.9156   3rd Qu.:0.9644   3rd Qu.:0.9467   3rd Qu.:1.0000
## Max.   :0.9511   Max.   :0.9911   Max.   :0.9911   Max.   :1.0000
```

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

```
## [1] 0.02519406 0.02091872 0.02726102 0.00318213
```

```
summary(TestAccuracy)
```

```
##        V1               V2               V3               V4
## Min.   :0.6133   Min.   :0.7200   Min.   :0.6667   Min.   :0.8533
## 1st Qu.:0.6933   1st Qu.:0.7733   1st Qu.:0.7467   1st Qu.:0.9067
## Median :0.7333   Median :0.8000   Median :0.7600   Median :0.9200
## Mean   :0.7232   Mean   :0.7983   Mean   :0.7669   Mean   :0.9199
## 3rd Qu.:0.7467   3rd Qu.:0.8267   3rd Qu.:0.7900   3rd Qu.:0.9333
## Max.   :0.8400   Max.   :0.8933   Max.   :0.8400   Max.   :0.9867
```

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

```
## [1] 0.04525007 0.03760053 0.03895285 0.02902037
```

```
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.   :0.8379  Min.   :0.8977  Min.   :0.8696  Min.   :0.9868
##  1st Qu.:0.8837  1st Qu.:0.9336  1st Qu.:0.9042  1st Qu.:0.9956
##  Median :0.9006  Median :0.9518  Median :0.9262  Median :0.9956
##  Mean   :0.8992  Mean   :0.9507  Mean   :0.9251  Mean   :0.9971
##  3rd Qu.:0.9168  3rd Qu.:0.9650  3rd Qu.:0.9467  3rd Qu.:1.0000
##  Max.   :0.9517  Max.   :0.9912  Max.   :0.9913  Max.   :1.0000
```

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

```
## [1] 0.024015157 0.020658595 0.026074580 0.003143959
```

```
summary(TestPrecisionMacroAve)
```

```
##       V1              V2              V3              V4
##  Min.   :0.6210  Min.   :0.7200  Min.   :0.6809  Min.   :0.8538
##  1st Qu.:0.7031  1st Qu.:0.7774  1st Qu.:0.7498  1st Qu.:0.9075
##  Median :0.7362  Median :0.8008  Median :0.7726  Median :0.9279
##  Mean   :0.7310  Mean   :0.8031  Mean   :0.7728  Mean   :0.9236
##  3rd Qu.:0.7594  3rd Qu.:0.8274  3rd Qu.:0.8005  3rd Qu.:0.9390
##  Max.   :0.8455  Max.   :0.8935  Max.   :0.8633  Max.   :0.9872
```

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

```
## [1] 0.04627204 0.03693013 0.03996421 0.02781195
```

```
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.   :0.8356  Min.   :0.8933  Min.   :0.8622  Min.   :0.9867
##  1st Qu.:0.8800  1st Qu.:0.9333  1st Qu.:0.9022  1st Qu.:0.9956
##  Median :0.8978  Median :0.9511  Median :0.9222  Median :0.9956
##  Mean   :0.8966  Mean   :0.9501  Mean   :0.9225  Mean   :0.9971
##  3rd Qu.:0.9156  3rd Qu.:0.9644  3rd Qu.:0.9467  3rd Qu.:1.0000
##  Max.   :0.9511  Max.   :0.9911  Max.   :0.9911  Max.   :1.0000
```

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

```
## [1] 0.02519406 0.02091872 0.02726102 0.00318213
```

```
summary(TestRecallMacroAve)
```

```
##       V1              V2              V3              V4
##  Min.   :0.6133  Min.   :0.7200  Min.   :0.6667  Min.   :0.8533
##  1st Qu.:0.6933  1st Qu.:0.7733  1st Qu.:0.7467  1st Qu.:0.9067
##  Median :0.7333  Median :0.8000  Median :0.7600  Median :0.9200
##  Mean   :0.7232  Mean   :0.7983  Mean   :0.7669  Mean   :0.9199
##  3rd Qu.:0.7467  3rd Qu.:0.8267  3rd Qu.:0.7900  3rd Qu.:0.9333
##  Max.   :0.8400  Max.   :0.8933  Max.   :0.8400  Max.   :0.9867
```

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

```
## [1] 0.04525007 0.03760053 0.03895285 0.02902037
```

```
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.   :0.8340   Min.   :0.8928   Min.   :0.8628   Min.   :0.9867
##  1st Qu.:0.8797   1st Qu.:0.9334   1st Qu.:0.9019   1st Qu.:0.9956
##  Median :0.8975   Median :0.9512   Median :0.9225   Median :0.9956
##  Mean   :0.8963   Mean   :0.9501   Mean   :0.9227   Mean   :0.9971
##  3rd Qu.:0.9158   3rd Qu.:0.9644   3rd Qu.:0.9465   3rd Qu.:1.0000
##  Max.   :0.9512   Max.   :0.9911   Max.   :0.9911   Max.   :1.0000
```

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

```
## [1] 0.025436534 0.020931430 0.027183454 0.003184041
```

```
summary(TestF1ScoreMacroAve)
```

```
##        V1               V2               V3               V4
##  Min.   :0.6134   Min.   :0.7199   Min.   :0.6689   Min.   :0.8517
##  1st Qu.:0.6902   1st Qu.:0.7729   1st Qu.:0.7429   1st Qu.:0.9061
##  Median :0.7283   Median :0.7987   Median :0.7611   Median :0.9203
##  Mean   :0.7206   Mean   :0.7978   Mean   :0.7660   Mean   :0.9196
##  3rd Qu.:0.7467   3rd Qu.:0.8258   3rd Qu.:0.7902   3rd Qu.:0.9336
##  Max.   :0.8397   Max.   :0.8927   Max.   :0.8411   Max.   :0.9867
```

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

```
## [1] 0.04561495 0.03753781 0.03938696 0.02929370
```

```
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"))
```