Projet2_scoring

Youming_Quantin

2023-10-30

Consigne

Les données : http://alexandrelourme.free.fr/M2IREF/SCORING/BANKNOTE sont composées de billets de banque répartis en deux classes : genuine/ counterfeit et décrits par : Longueur Bord Gauche (Left), Largeur Bord Droit (Right). On envisage trois méthodes de scoring : AD Gaussienne Hétéroscédastique, AD Gaussienne Homoscédastique, Resgression Logistrique pour affecter ç une classe existante un nouveau billet caractérisé par :

- La class de nouveau billet : Length = 214,90 Right = 129,96 Left = 130,12

EX 1. A quelle classe chacune de ces trois méthodes affecte-t-elle le nouveau billet ?

```
rm(list=ls(all=TRUE))
library(stats)
library(Rmixmod)
```

```
## Loading required package: Rcpp
```

```
## Rmixmod v. 2.1.8 / URI: www.mixmod.org
```

```
x = read.table(file = "http://alexandrelourme.free.fr/M2IREF/SCORING/BANKNOTE" , sep=',', dec='.'
table(x$Status)
```

```
##
## counterfeit     genuine
##         100         100
```

```
#counterfeit     genuine
#        100         100
```

```
train1 <- x[,2:4]
ztrain <- x[,1]
```

```r
head(ztrain)
```

```
## [1] "genuine" "genuine" "genuine" "genuine" "genuine" "genuine"
```

EX1

Gaussienne Hétéroscédastique

```r
learn <- mixmodLearn(x[,2:4],knownLabels = as.factor(x[,1]), models = mixmodGaussianModel(listMode
learn
```

```
## ****************************************
## *** INPUT:
## ****************************************
## * nbCluster =  2
## * criterion =  BIC
## ****************************************
## *** MIXMOD Models:
## * list =  Gaussian_pk_Lk_Ck
## * This list includes only models with free proportions.
## ****************************************
## * data (limited to a 10x10 matrix) =
##       Length Left  Right
##  [1,] 214.8  131   131.1
##  [2,] 214.6  129.7 129.7
##  [3,] 214.8  129.7 129.7
##  [4,] 214.8  129.7 129.6
##  [5,] 215    129.6 129.7
##  [6,] 215.7  130.8 130.5
##  [7,] 215.5  129.5 129.7
##  [8,] 214.5  129.6 129.2
##  [9,] 214.9  129.4 129.7
## [10,] 215.2  130.4 130.3
## * ... ...
## * knownLabels =  2 2 2 2 2 2 2 2 2 2  ...
##
##
## ****************************************
## *** BEST MODEL OUTPUT:
## *** According to the BIC criterion
## ****************************************
```

```
## * nbCluster   =  2
## * model name  =  Gaussian_pk_Lk_Ck
## * criterion   =  BIC(610.3037)
## * likelihood  =  -254.8178
## ****************************************
## *** Cluster 1
## * proportion =  0.5000
## * means      =  214.8230 130.3000 130.1930
## * variances  = |     0.1228     0.0312     0.0238 |
##                |     0.0312     0.0644     0.0463 |
##                |     0.0238     0.0463     0.0881 |
## *** Cluster 2
## * proportion =  0.5000
## * means      =  214.9690 129.9430 129.7200
## * variances  = |     0.1487     0.0574     0.0567 |
##                |     0.0574     0.1313     0.0850 |
##                |     0.0567     0.0850     0.1250 |
## ****************************************
## * Classification with MAP:
##            | Cluster 1 | Cluster 2 |
## ---------- ----------- -----------
##  Cluster 1 |       100 |         0 |
##  Cluster 2 |         0 |       100 |
## ---------- ----------- -----------
## * Error rate with MAP =  0.00 %
## ****************************************
```

```
new = as.data.frame(t(c(214.9, 130.12, 129.96))) # nouveau individue
prédiction <- mixmodPredict(data = new, classificationRule = learn['bestResult'])
prédiction
```

```
## ****************************************
## *** INPUT:
## ****************************************
## * nbCluster   =  2
## * model name  =  Gaussian_pk_Lk_Ck
## * criterion   =  BIC(610.3037)
## * likelihood  =  -254.8178
## ****************************************
## *** Cluster 1
## * proportion =  0.5000
```

```
## * means      =  214.8230 130.3000 130.1930
## * variances  = |     0.1228     0.0312     0.0238 |
##                |     0.0312     0.0644     0.0463 |
##                |     0.0238     0.0463     0.0881 |
## *** Cluster 2
## * proportion =  0.5000
## * means      =  214.9690 129.9430 129.7200
## * variances  = |     0.1487     0.0574     0.0567 |
##                |     0.0574     0.1313     0.0850 |
##                |     0.0567     0.0850     0.1250 |
## ****************************************
## * Classification with MAP:
##            | Cluster 1 | Cluster 2 |
## ---------- ---------- ----------
##  Cluster 1 |       100 |         0 |
##  Cluster 2 |         0 |       100 |
## ---------- ---------- ----------
## * Error rate with MAP =  0.00 %
## ****************************************
## * data (limited to a 10x10 matrix) =
##    V1    V2    V3
## 214.9 130.1   130
## * ... ...
##
##
## ****************************************
## *** PREDICTION:
## ****************************************
## * partition     =  1
## * probabilities =  0.6028 0.3972
## ****************************************
```

Elle est dans la classe 1, $t_1 = 0.6028$ et $t_2 = 0.3972$

```
x1 <- x[x[, 1] == "genuine", ]
barx1 <- t(colMeans(x1[,2:4]))
x2 <- x[x[, 1] == "counterfeit", ]
barx2 <- t(colMeans(x2[,2:4]))
print(barx1)
```

```
##       Length    Left  Right
```

```
## [1,] 214.969 129.943 129.72
```

```
print(barx2)
```

```
##       Length  Left   Right
## [1,] 214.823 130.3 130.193
```

La cluster 1 est **counterfeit**, qui a une moyenne corérent :214.823 130.3 130.193 La cluster 2 est **genuine**, qui a une moyenne corérent : 214.969 129.943 129.72 L'AD Gaussienne hétéroscédastique (Gaussian_pk_Lk_Ck) affecte la nouvelle individue à la classe **counterfeit**

Gaussienne Homoscédastique

```
learn <- mixmodLearn(x[,2:4],knownLabels = as.factor(x[,1]), models = mixmodGaussianModel(listMode
learn
```

```
## ****************************************
## *** INPUT:
## ****************************************
## * nbCluster =  2
## * criterion =  BIC
## ****************************************
## *** MIXMOD Models:
## * list =  Gaussian_pk_L_C
## * This list includes only models with free proportions.
## ****************************************
## * data (limited to a 10x10 matrix) =
##       Length Left  Right
##  [1,] 214.8  131   131.1
##  [2,] 214.6  129.7 129.7
##  [3,] 214.8  129.7 129.7
##  [4,] 214.8  129.7 129.6
##  [5,] 215    129.6 129.7
##  [6,] 215.7  130.8 130.5
##  [7,] 215.5  129.5 129.7
##  [8,] 214.5  129.6 129.2
##  [9,] 214.9  129.4 129.7
## [10,] 215.2  130.4 130.3
## * ... ...
## * knownLabels =  2 2 2 2 2 2 2 2 2 2  ...
##
##
```

```
## ****************************************
## *** BEST MODEL OUTPUT:
## *** According to the BIC criterion
## ****************************************
## * nbCluster   =  2
## * model name  =  Gaussian_pk_L_C
## * criterion   =  BIC(594.7856)
## * likelihood  =  -262.9538
## ****************************************
## *** Cluster 1
## * proportion =  0.5000
## * means      =  214.8230 130.3000 130.1930
## * variances  = |     0.1358     0.0443     0.0402 |
##                |     0.0443     0.0978     0.0657 |
##                |     0.0402     0.0657     0.1065 |
## *** Cluster 2
## * proportion =  0.5000
## * means      =  214.9690 129.9430 129.7200
## * variances  = |     0.1358     0.0443     0.0402 |
##                |     0.0443     0.0978     0.0657 |
##                |     0.0402     0.0657     0.1065 |
## ****************************************
## * Classification with MAP:
##            | Cluster 1 | Cluster 2 |
## ---------- ---------- ----------
##  Cluster 1 |       100 |         0 |
##  Cluster 2 |         0 |       100 |
## ---------- ---------- ----------
## * Error rate with MAP =  0.00 %
## ****************************************
```

```
new = as.data.frame(t(c(214.9, 130.12, 129.96))) # nouveau individue
prédiction <- mixmodPredict(data = new, classificationRule = learn['bestResult'])
prédiction
```

```
## ****************************************
## *** INPUT:
## ****************************************
## * nbCluster   =  2
## * model name  =  Gaussian_pk_L_C
## * criterion   =  BIC(594.7856)
```

```
## * likelihood  =  -262.9538
## ****************************************
## *** Cluster 1
## * proportion =  0.5000
## * means      =  214.8230 130.3000 130.1930
## * variances  = |     0.1358     0.0443     0.0402 |
##                |     0.0443     0.0978     0.0657 |
##                |     0.0402     0.0657     0.1065 |
## *** Cluster 2
## * proportion =  0.5000
## * means      =  214.9690 129.9430 129.7200
## * variances  = |     0.1358     0.0443     0.0402 |
##                |     0.0443     0.0978     0.0657 |
##                |     0.0402     0.0657     0.1065 |
## ****************************************
## * Classification with MAP:
##            | Cluster 1 | Cluster 2 |
## ---------- ----------- -----------
##  Cluster 1 |       100 |         0 |
##  Cluster 2 |         0 |       100 |
## ---------- ----------- -----------
## * Error rate with MAP =  0.00 %
## ****************************************
## * data (limited to a 10x10 matrix) =
##    V1    V2    V3
## 214.9 130.1   130
## * ... ...
##
##
## ****************************************
## *** PREDICTION:
## ****************************************
## * partition     =  2
## * probabilities =  0.4999 0.5001
## ****************************************
```

Elle est dans la classe 2, $t_1 = 0.4999$ et $t_2 = 0.5001$ La cluster 1 est **counterfeit**, qui a une moyenne corérent :214.823 130.3 130.193 La cluster 2 est **genuine**, qui a une moyenne corérent : 214.969 129.943 129.72 L'AD Gaussienne Homoscédastique (Gaussian_pk_L_C) affecte la nouvelle individue à la classe **genuine**

	Proba Bankcuryty healty	BIC
Gaussien Homoscédastique	0.4999 vs 0.5001	594/2
Gaussien Hétéscédastique	0.6028 vs 0.3972	610/2

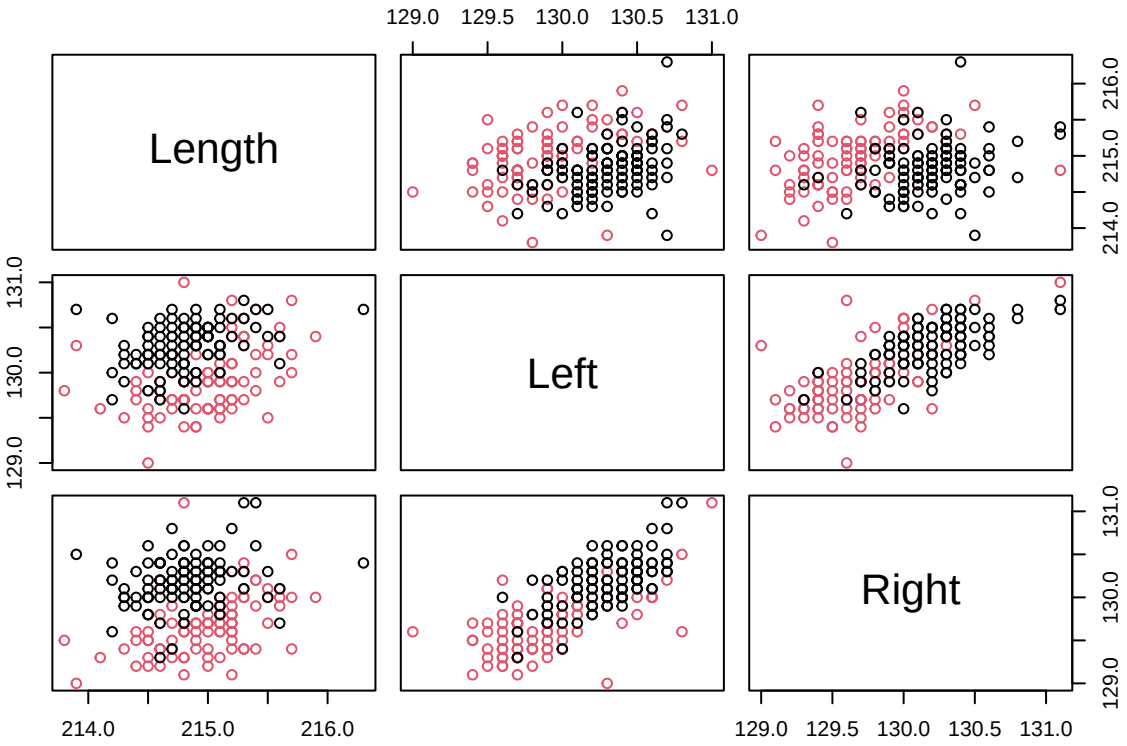
S'il faut choisir un, on choisit le BIC le plus élevé

Regression logistique

```
attach(x) # train data

# Test data
test = data.frame(Length = 214.9, Left = 130.12, Right = 129.96) # test data

plot(x[,2:4], col = as.factor(x[,1]))
```




```
# Logistic Regression with glm
rule = glm(as.factor(Status) ~ Length + Left + Right, data = x, family = binomial(link = 'logit'))

# Coefficients estimation
alpha = rule$coefficients[1]
beta_1 = rule$coefficients[2]
beta_2 = rule$coefficients[3]
```

```r
beta_3 = rule$coefficients[3]
```

```r
score <- predict(rule,new=test) ; print(score)
```

```
##          1
## 0.01562366
```

La valeur :

$$\beta_1 \times Length + \beta_2 \times Left + \beta_3 \times Right + \alpha$$

calculée dans le test, #0.01562366, est positive, donc le nouveau billet est dans la classe 1.

```r
prob <- exp(score)/(1+exp(score)) ; print(prob) # probability of belonging to Class 1, qui est > 0
```

```
##         1
## 0.5039058
```

Ce résultat signifie la probabilité qu'un individu soit affecté par le cluster 1. On voit il est supérieur à 0.5, donc le nouveau billet est dabs la classe 1

EX2 Déterminer l'erreur apparente de chacun des trois classifieurs

$$\hat{\varepsilon} = \#\{i; \hat{y}(x_i) \neq y_i\}/n$$

```r
# une fonction qui calcule l'erreur de classement du classifieur gaussien homoscédastique
ergauss <- function(train,trainlab,test,testlab){
  trainlab=as.factor(trainlab)
  testlab=as.factor(testlab)
  yourmodel=mixmodGaussianModel(listModels=c("Gaussian_pk_L_C"))
  out <- mixmodLearn(train, knownLabels=trainlab, models = yourmodel)
  pred <- mixmodPredict(data = test, classificationRule = out["bestResult"])
  erg <- sum(pred[5]!=as.numeric(testlab))/length(testlab) # taux d'erreur sur les données de t
}
############################
# estimation de l'erreur par resubstitution (erreur apparente)
############################
epsilon_ap=ergauss(train1,ztrain,train1,ztrain)
cat('epsilon_ap_homoscédastique :',epsilon_ap,'\n')
```

```
## epsilon_ap_homoscédastique : 0.18
```

```r
## epsilon_ap_homoscédastique : 0.18

# une fonction qui calcule l'erreur de classement du classifieur gaussien hétéroscédastique
```

```
ergausshete <- function(train,trainlab,test,testlab){
  trainlab=as.factor(trainlab)
  testlab=as.factor(testlab)
  yourmodel=mixmodGaussianModel(listModels=c("Gaussian_pk_Lk_Ck"))
  out <- mixmodLearn(train, knownLabels=trainlab, models = yourmodel)
  pred <- mixmodPredict(data = test, classificationRule = out["bestResult"])
  erg = sum(pred[5]!=as.numeric(testlab))/length(testlab) # taux d'erreur sur les données de t
  return(erg)
}
############################
# estimation de l'erreur par resubstitution (erreur apparente)
############################
epsilon_ap=ergausshete(train1,ztrain,train1,ztrain)
cat('epsilon_ap_hétéroscédastique :',epsilon_ap,'\n')
```

```
## epsilon_ap_hétéroscédastique : 0.155
```

```
## epsilon_ap_hétéroscédastique : 0.155

# une fonction qui calcule l'erreur de classement du classifieur regression logistique

Z=matrix(0,nrow=200,ncol=1) # tableau disjonctif complet/matrice des appartenances
for (i in 1:200){
  Z[i,1]=ifelse (x[i,1]=="genuine",0,1)
  Z[i,1]=ifelse (x[i,1]=="counterfeit",0,1)
}

logistique <- function(train,trainlab,test, testlab){
  yourmodel='logit'
  train_column_names <- names(train)
  formula_string <- paste("as.factor(trainlab) ~", paste(train_column_names, collapse = " + "))
  formula <- as.formula(formula_string)
  rule <- glm(formula, data = train, family = binomial(link = yourmodel))
  predicted_probabilities <- predict(rule,test, type = "response")
  predicted_classes <- ifelse(predicted_probabilities > 0.5, 1, 0)
  rapport <- table(as.factor(testlab) == predicted_classes)
  epsilon_ap=rapport[1]/length(testlab)
}

epsilon_ap=logistique(train1,ztrain,x,Z)
```

```r
cat('epsilon_ap_logistique :',epsilon_ap,'\n')
```

```
## epsilon_ap_logistique : 0.18
```

```r
#epsilon_ap_logistique : 0.18
```

##EX3 Déterminer l'erreur de Validation Croisé de chacun des trois classifieurs

```r
##############################################
# estimation de l'erreur par v-fold Cross Validation pour Gaussian homoscédastique
##############################################
K=5 # nbre de sous échantillons
epsilonlist <- NULL
n <- nrow(x)
list=sample(1:n,n,replace=FALSE)
for (j in 1:K){
  train <- x[-c(((2*j-1):(2*j))),] # échantillon d'apprentissage
  test <- x[c(((2*j-1):(2*j))),] # échantillon de test
  epsilonlist[j] <- ergauss(train1,ztrain,train1,ztrain)
}
epsilon_vfoldcv = mean(epsilonlist)
cat('epsilon_vfoldcv_homoscédastique :',epsilon_vfoldcv,'\n')
```

```
## epsilon_vfoldcv_homoscédastique : 0.18
```

```r
## epsilon_vfoldcv_homoscédastique : 0.18



##############################################
# estimation de l'erreur par v-fold Cross Validation pour Gaussian Hétéroscédastique
##############################################

K=5 # nbre de sous échantillons
epsilonlist <- NULL
n <- nrow(x)
list=sample(1:n,n,replace=FALSE)
for (j in 1:K){
  train <- x[-c(((2*j-1):(2*j))),] # échantillon d'apprentissage
  test <- x[c(((2*j-1):(2*j))),] # échantillon de test
  epsilonlist[j] <- ergausshete(train1,ztrain,train1,ztrain)
}
epsilon_vfoldcv = mean(epsilonlist)
```

```r
cat('epsilon_vfoldcv_hétéroscédastique :',epsilon_vfoldcv,'\n')
```

```
## epsilon_vfoldcv_hétéroscédastique : 0.155
```

```r
## epsilon_vfoldcv_hétéroscédastique : 0.155
```

```r
##############################################
# estimation de l'erreur par v-fold Cross Validation pour régression logistique
##############################################
K=5 # nbre de sous échantillons
epsilonlist <- NULL
n <- nrow(x)
list=sample(1:n,n,replace=FALSE)
for (j in 1:K){
  train <- x[-c(((2*j-1):(2*j))),] # échantillon d'apprentissage
  test <- x[c(((2*j-1):(2*j))),] # échantillon de test
  epsilonlist[j] <- logistique(train1,ztrain,x,Z)
}
epsilon_vfoldcv = mean(epsilonlist)
cat('epsilon_vfoldcv_logistique :',epsilon_vfoldcv,'\n')
```

```
## epsilon_vfoldcv_logistique : 0.18
```

```r
## epsilon_vfoldcv_logistique: 0.18
```

EX4 Tracer la courbe ROC du classifieur Gaussien homoscédastique

```r
plotROC <- function(train, trainlab, test, testlab) {
  train_lab <- as.factor(trainlab)
  testlab <- as.factor(testlab)
  mymodel <- mixmodGaussianModel(listModels = c("Gaussian_pk_L_C"))
  out <- mixmodLearn(train, knownLabels = train_lab, models = mymodel)
  pred <- mixmodPredict(data = test, classificationRule = out["bestResult"])
  t <- data.frame(pred[6])
  colnames(t) <- c("t1","t2") #obtenir les probabilités

  compare <- data.frame(testlab = testlab)
#  unique_categories <- unique(factor(compare$testlab))
#  category_mapping <- setNames(1:length(unique_categories), unique_categories)
```

```r
#  categories_numeric <- as.numeric(categories_factor)
#  compare$testlab <- as.numeric(category_mapping)
#  compare$decision <- compare$testlab
  categories_factor <- factor(compare$testlab)
  unique_categories <- unique(categories_factor)
  category_mapping <- setNames(1:length(unique_categories), unique_categories)
  compare$testlab <- as.numeric(categories_factor)
    TFP <- TVP <- NULL

  for (i in 0:nrow(test)) {
    compare$decision[t$t2 > i / nrow(test)] <- 2
    compare$decision[t$t2 <= i / nrow(test)] <- 1
    testlab <- factor(compare$testlab, levels = c(1, 2))
    decision <- factor(compare$decision, levels = c(1, 2))
    confusion <- table(testlab, decision)
    TFP[i] <- confusion[1, 2] / (confusion[1, 1] + confusion[1, 2])
    TVP[i] <- confusion[2, 2] / (confusion[2, 1] + confusion[2, 2])
  }
  plot(TFP, TVP, type = "l", xlab = "TFP", ylab = "TVP")
}
plotROC(train1,ztrain,train1,ztrain)
```