R Notebook

# Chargement des donnees nettoyees

dirpath <- "C:/Users/Eric/Documents/Eric/Pro/Transition/Formation/CEPE ENSAE ENSAI Certificat Data Scientist/INTENSIVE/PROJETS/KickClub Project"  
dirpath2 <- "Lending Club/LC0715"  
filename <- "loan5.RDS"  
my\_file <- paste(dirpath, dirpath2, filename, sep = "/")  
my\_file

## [1] "C:/Users/Eric/Documents/Eric/Pro/Transition/Formation/CEPE ENSAE ENSAI Certificat Data Scientist/INTENSIVE/PROJETS/KickClub Project/Lending Club/LC0715/loan5.RDS"

dta <- readRDS(my\_file)

# Menu des modélisations

1. Logistic Regression
2. LDA Linear Discriminant Analysis
3. QDA Quadratic Discriminant Analysis
4. KNN
5. Classification Trees
6. Random Forests

# # 1. Logistic Regression

# Extension de la matrice de confusion: Creer une fonction ‘Table.Ratio’ qui calcule les % de reussite par ligne et par colonne

Table.Ratio creates a matrix that expands the standard table with % of correctly predicted vs. actual observations for each row and each column

Table.Ratio <- function(predicted, actual){  
 table.res <- as.matrix(table(predicted, actual))  
 r1 <- table.res[1,1] / (table.res[1,1] + table.res[1,2])  
 r2 <- table.res[2,2] / (table.res[2,1] + table.res[2,2])  
 tot <- (table.res[1,1] + table.res[2,2]) / (table.res[1,1] + table.res[1,2] + table.res[2,1] + table.res[2,2])  
 c1 <- table.res[1,1] / (table.res[1,1] + table.res[2,1])  
 c2 <- table.res[2,2] / (table.res[1,2] + table.res[2,2])  
 newrow <- c(c1, c2)  
 newcol <- c(r1, r2, tot)  
 table.res <- rbind(table.res, col\_ratio = newrow)  
 table.res <- cbind(table.res, row\_ratio = newcol)  
 table.res <- round(table.res,2)  
 # rm(r1, r2, tot, c1, c2, newrow, newcol)  
 return(table.res)  
}

# Selection aleatoire d’un echantillon de test (In-Sample random ‘train’ subset)

In-Sample: 80% des observations

train <- sample(dim(dta)[1], size = 0.8 \* dim(dta)[1], replace = FALSE)

# Faire tourner la regression loogistique

glm.fit <- glm(loan\_status ~ . , data = dta[train,], family = binomial)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# Interpretation de regression logistique

Comment sont alloues les resultats:

contrasts(dta$loan\_status)

## Fully Paid  
## Charged Off 0  
## Fully Paid 1

# Resultats Out-Of-Sample

Previsions de Probabilites issues de la regression logistique sur l’echantillon de validation

glm.probs <- predict(glm.fit, newdata = dta[-train,], type = "response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

Creation d’un vecteur de previsions selon le niveau de probabilite

glm.pred <- rep("Valid", dim(dta[-train,])[1]) # initialisation  
glm.pred[glm.probs < quantile(glm.probs, probs = 0.5)] <- "Failed" # le seuil est fixe au niveau du 50eme percentile des previsions

Matrice de Confusion etendue: previsions en lignes (Failed et Valid) et realises en colonnes (Charged Off et Fully Paid)

Table.Ratio(glm.pred, dta[-train,]$loan\_status)

## Charged Off Fully Paid row\_ratio  
## Failed 6762.00 18497.00 0.27  
## Valid 2279.00 22980.00 0.91  
## col\_ratio 0.75 0.55 0.59

# Creation d’une fonction Run.n.random.logreg

* Pour realiser ‘n’ regressions logistiques
* Sur le dataset ‘dta’
* Et sur le subset in-sample ‘train’
* Et, qui renvoie la pertinence globale de la matrice de confusion des previsions

Run.n.random.logreg <- function(n = 2, dta, insample\_prop = 0.8) {  
 res <- matrix(, nrow = n, ncol = 1)  
 for (i in 1:n) {  
 train <- sample(dim(dta)[1], size = insample\_prop \* dim(dta)[1], replace = FALSE)  
 glm.fit <- glm(loan\_status ~ . , data = dta[train,], family = binomial)  
 glm.probs <- predict(glm.fit, newdata = dta[-train,], type = "response")  
 glm.pred <- rep("Valid", dim(dta[-train,])[1])  
 glm.pred[glm.probs < quantile(glm.probs, probs = 0.5)] <- "Failed"  
 res[i,1] <- Table.Ratio(glm.pred, dta[-train,]$loan\_status)[3,3]  
 }  
 return(res)   
}

# Application: n = 20 regressions logistiques, insample\_prop = 80%

Attention: cela prend un peu de temps avec 20 regressions logistiques (envisager le calcul parallele!?)

Run.n.random.logreg(n = 20, dta, insample\_prop = 0.8)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## [,1]  
## [1,] 0.59  
## [2,] 0.59  
## [3,] 0.59  
## [4,] 0.59  
## [5,] 0.59  
## [6,] 0.59  
## [7,] 0.59  
## [8,] 0.59  
## [9,] 0.59  
## [10,] 0.59  
## [11,] 0.59  
## [12,] 0.59  
## [13,] 0.59  
## [14,] 0.59  
## [15,] 0.59  
## [16,] 0.59  
## [17,] 0.59  
## [18,] 0.59  
## [19,] 0.59  
## [20,] 0.59

# Nettoyage de l’environnement

rm(glm.fit, glm.probs, glm.pred, train)

# # 2. Linear Discriminant Analysis - LDA

# Librairie ‘MASS’

library(MASS)

# Selection aleatoire d’un echantillon de test (In-Sample random ‘train’ subset)

In-Sample: 80% des observations

train <- sample(dim(dta)[1], size = 0.8 \* dim(dta)[1], replace = FALSE)

# Faire tourner une LDA Run LDA sur l’echantillon test (‘train’)

lda.fit <- lda(loan\_status ~ . , data = dta, subset = train) # In lda.default(x, grouping, ...) : les variables sont collinéaires

## Warning in lda.default(x, grouping, ...): variables are collinear

# Generer les previsions de probabilites du modele LDA sur l’echantillon de validation

lda.pred <- predict(lda.fit, dta[-train,])  
names(lda.pred)

## [1] "class" "posterior" "x"

# ‘posterior’ contient les previsions de probabilites generees par le modele LDA

mean(lda.pred$posterior) # la prevision moyenne est de 50% comme attendu

## [1] 0.5

summary(lda.pred$posterior)

## Charged Off Fully Paid   
## Min. :0.002582 Min. :0.08794   
## 1st Qu.:0.091536 1st Qu.:0.76695   
## Median :0.142853 Median :0.85715   
## Mean :0.179512 Mean :0.82049   
## 3rd Qu.:0.233051 3rd Qu.:0.90846   
## Max. :0.912058 Max. :0.99742

La prevision moyenne et mediane du pret ‘Charged Off’ est 17.9% et 14.2% Inversement pour les prets ‘Fully Paid’: 82.1% et 85.8% Attention: ces moyennes et medianes varient en fonction de l’echantillon aleatoire Conclusion: il faut adapter le seuil d’allocation. Un seuil de 50% prevoirait trop de defaut!

# ‘class’ contient l’allocation standard des previsions par modalite au seuil de proababilite de base de 50%

lda.class <- lda.pred$class  
lda.class[1:30] # jetez un oeil aux previsions au seuil standard de 50%

## [1] Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid  
## [7] Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid  
## [13] Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid  
## [19] Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid  
## [25] Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid  
## Levels: Charged Off Fully Paid

Il y a tres peu de previsions de ‘Charged Off’ au seuil de 50% de probabilite

# Creation d’une matrice de confusion: previsions vs. realisations

table(lda.class, dta[-train,]$loan\_status)

##   
## lda.class Charged Off Fully Paid  
## Charged Off 670 682  
## Fully Paid 8360 40806

Et sa version etendue avec les % en utilisant Table.Ratio() creee auparavant

Table.Ratio(lda.class, dta[-train,]$loan\_status)

## Charged Off Fully Paid row\_ratio  
## Charged Off 670.00 682.00 0.50  
## Fully Paid 8360.00 40806.00 0.83  
## col\_ratio 0.07 0.98 0.82

Cela peut paraitre eleve au global (~82%), mais ce resultat est decevant: - seuls 8% des ‘Charged Off’ sont predit ‘Charge Off’ par le modele - seuls 50% des predictions de ‘Charge Off’ sont effectivement ‘Charged Off’

# Creer une fonction qui ajuste le seuil de probabilite et genere une matrice de confusion

Notez: la LDA est deja realisee. Il s’agit de moduler le seuil d’allocation des previsions

Confusion.Table <- function(actual, predicted, proba.threshold){  
# actual: the actual observed dataset  
# lda.pred: the predicted probabilities based on LDA model  
# proba.threshold: the threshold used to assign probabilities to each outcome  
 actual.Failed\_pred <- actual[predicted > proba.threshold]  
 actual.Valid\_pred <- actual[predicted <= proba.threshold]  
 Confusion.Table <- matrix(1:9, nrow = 3)  
 Confusion.Table[1,1] <- table(actual.Failed\_pred)[1]  
 Confusion.Table[1,2] <- table(actual.Failed\_pred)[2]  
 Confusion.Table[2,1] <- table(actual.Valid\_pred)[1]  
 Confusion.Table[2,2] <- table(actual.Valid\_pred)[2]  
 colnames(Confusion.Table) <- c("Actual.Failed", "Actual.Valid", "Accuracy")  
 rownames(Confusion.Table) <- c("Pred.Failed", "Pred.Valid", "Accuracy")  
 Confusion.Table[1,3] <- Confusion.Table[1,1] / (Confusion.Table[1,1] + Confusion.Table[1,2])  
 Confusion.Table[2,3] <- Confusion.Table[2,2] / (Confusion.Table[2,1] + Confusion.Table[2,2])  
 Confusion.Table[3,1] <- Confusion.Table[1,1] / (Confusion.Table[1,1] + Confusion.Table[2,1])  
 Confusion.Table[3,2] <- Confusion.Table[2,2] / (Confusion.Table[1,2] + Confusion.Table[2,2])  
 Confusion.Table[3,3] <- (Confusion.Table[1,1] + Confusion.Table[2,2]) / sum(Confusion.Table[1:2, 1:2])  
 return(Confusion.Table)  
}

Mise en application pas à pas

Mise en oeuvre de la fonction : etape par etape (indirecte): on prend soin de definir les parametre un a un

actual <- dta[-train, "loan\_status"] # actual status over the 'train' subset  
pred <- lda.pred$posterior[,1] # predicted probabilities (1st column = 'Failed') from LDA model  
proba.threshold <- mean(pred) # average predicted probability instead of 50% probability threshold  
Confusion.Table(actual, pred, proba.threshold)

## Actual.Failed Actual.Valid Accuracy  
## Pred.Failed 5651.0000000 1.325700e+04 0.2988682  
## Pred.Valid 3379.0000000 2.823100e+04 0.8931034  
## Accuracy 0.6258029 6.804618e-01 0.6706916

round(Confusion.Table(actual, pred, proba.threshold),2)

## Actual.Failed Actual.Valid Accuracy  
## Pred.Failed 5651.00 13257.00 0.30  
## Pred.Valid 3379.00 28231.00 0.89  
## Accuracy 0.63 0.68 0.67

Mise ne oeuvre directe: les parametres sont definis a l’interieur au risque d’etre moins visible

round(Confusion.Table(dta[-train, "loan\_status"], lda.pred$posterior[,1], mean(lda.pred$posterior[,1])),2)

## Actual.Failed Actual.Valid Accuracy  
## Pred.Failed 5651.00 13257.00 0.30  
## Pred.Valid 3379.00 28231.00 0.89  
## Accuracy 0.63 0.68 0.67

Commentaire: - le niveau global de prevision s’est deteriore (67%) - mais la qualite de prevision des defauts effectifs s’est fortement ameliore: on prevoit mieux les vrais defauts (62%) - et les previsions de defauts ont acrru en nombre mais ont perdu en qualite predictive (30%)

Nettoyage

rm(train, lda.class, lda.pred, lda.fit)

# Creation d’une fonction Run.n.random.ldareg

* Pour realiser ‘n’ regressions lineaires discriminantes
* Sur le dataset ‘dta’
* Et sur le subset in-sample ‘train’ de proportion ‘insample\_prop’
* Et, qui renvoie la pertinence globale de la matrice de confusion des previsions

Run.n.random.ldareg <- function(n = 2, dta, insample\_prop = 0.8, threshold = c("mean", "median", "50%")) {  
 res <- matrix(, nrow = n, ncol = 5)  
 for (i in 1:n) {  
 train <- sample(dim(dta)[1], size = insample\_prop \* dim(dta)[1], replace = FALSE)  
 lda.fit <- lda(loan\_status ~ . , data = dta, subset = train)  
 lda.pred <- predict(lda.fit, dta[-train,])  
 lda.class <- lda.pred$class  
   
 actual <- dta[-train, "loan\_status"] # actual status over the 'train' subset  
 pred <- lda.pred$posterior[,1] # predicted probabilities (1st column = 'Failed') from LDA model  
 proba.threshold <- switch(threshold,  
 "mean" = mean(pred), # average predicted probability  
 "median" = median(pred), # median predicted probability  
 "50%" = 0.5) # 50% probability  
 Confusion.Table(actual, pred, proba.threshold)  
 res[i,1] <- round(Confusion.Table(actual, pred, proba.threshold)[3,3],2)  
 res[i,2] <- round(Confusion.Table(actual, pred, proba.threshold)[3,1],2)  
 res[i,3] <- round(Confusion.Table(actual, pred, proba.threshold)[1,3],2)  
 res[i,4] <- round(Confusion.Table(actual, pred, proba.threshold)[3,2],2)  
 res[i,5] <- round(Confusion.Table(actual, pred, proba.threshold)[2,3],2)  
 colnames(res) <- c("Global", "Actual.Failed", "Pred.Failed", "Actual.Valid", "Pred.Valid")  
   
 }  
 return(res)   
}

# Application: n = 10 regressions lineaires discriminantes, insample\_prop = 80%, threshold = “mean”

Attention: cela prend un peu de temps avec 10 regressions logistiques (envisager le calcul parallele!?)

Run.n.random.ldareg(n = 10, dta, insample\_prop = 0.8, threshold = "mean")

## Warning in lda.default(x, grouping, ...): variables are collinear  
  
## Warning in lda.default(x, grouping, ...): variables are collinear  
  
## Warning in lda.default(x, grouping, ...): variables are collinear  
  
## Warning in lda.default(x, grouping, ...): variables are collinear  
  
## Warning in lda.default(x, grouping, ...): variables are collinear  
  
## Warning in lda.default(x, grouping, ...): variables are collinear  
  
## Warning in lda.default(x, grouping, ...): variables are collinear  
  
## Warning in lda.default(x, grouping, ...): variables are collinear  
  
## Warning in lda.default(x, grouping, ...): variables are collinear  
  
## Warning in lda.default(x, grouping, ...): variables are collinear

## Global Actual.Failed Pred.Failed Actual.Valid Pred.Valid  
## [1,] 0.67 0.61 0.3 0.68 0.89  
## [2,] 0.67 0.63 0.3 0.68 0.89  
## [3,] 0.67 0.62 0.3 0.68 0.89  
## [4,] 0.67 0.63 0.3 0.68 0.89  
## [5,] 0.67 0.62 0.3 0.68 0.89  
## [6,] 0.67 0.62 0.3 0.68 0.89  
## [7,] 0.67 0.63 0.3 0.68 0.89  
## [8,] 0.67 0.62 0.3 0.68 0.89  
## [9,] 0.67 0.63 0.3 0.68 0.89  
## [10,] 0.67 0.63 0.3 0.68 0.89

# Nettoyage de l’environnement

rm(lda.fit, lda.pred, lda.class, train, pred, proba.threshold)

## Warning in rm(lda.fit, lda.pred, lda.class, train, pred, proba.threshold):  
## objet 'lda.fit' introuvable

## Warning in rm(lda.fit, lda.pred, lda.class, train, pred, proba.threshold):  
## objet 'lda.pred' introuvable

## Warning in rm(lda.fit, lda.pred, lda.class, train, pred, proba.threshold):  
## objet 'lda.class' introuvable

## Warning in rm(lda.fit, lda.pred, lda.class, train, pred, proba.threshold):  
## objet 'train' introuvable

# # 3. Quadratic Discriminant Analysis - QDA

# Librairie ‘MASS’

library(MASS)  
library(dplyr)

## Warning: package 'dplyr' was built under R version 3.4.3

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':  
##   
## select

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

# Selection aleatoire d’un echantillon de test (In-Sample random ‘train’ subset)

In-Sample: 80% des observations

train <- sample(dim(dta)[1], size = 0.8 \* dim(dta)[1], replace = FALSE)

# Exclusion additionnelle de variables qui font planter la QDA

# identification des variables problematiques: qui font planter la QDA: "groupe Charged Off n'est pas de rang plein"  
dta\_m <- dta %>% dplyr::select( c( names(dta)[1:6], names(dta)[8:12], names(dta)[14:25] ) )  
  
# Exclusion de ces variables: creation d'un subset 'dta\_m' à utiliser pour la suite de la QDA  
dta\_m <- dta %>% dplyr::select(-c(sub\_grade, addr\_state))

# Faire tourner une QDA sur l’echantillon test (‘train’)

qda.fit <- qda(loan\_status ~ . , data = dta\_m, subset = train) # In lda.default(x, grouping, ...) : les variables sont collinéaires

# Generer les previsions de probabilites du modele QDA sur l’echantillon de validation

qda.pred <- predict(qda.fit, dta\_m[-train,])  
names(qda.pred)

## [1] "class" "posterior"

# ‘posterior’ contient les previsions de probabilites generees par le modele QDA

mean(qda.pred$posterior) # la prevision moyenne est de 50% comme attendu

## [1] 0.5

summary(qda.pred$posterior)

## Charged Off Fully Paid   
## Min. :0.000000 Min. :0.0000   
## 1st Qu.:0.001726 1st Qu.:0.5752   
## Median :0.068798 Median :0.9312   
## Mean :0.262848 Mean :0.7372   
## 3rd Qu.:0.424839 3rd Qu.:0.9983   
## Max. :1.000000 Max. :1.0000

La prevision moyenne et mediane du pret ‘Charged Off’ est 23.7% et 4.6% Inversement pour les prets ‘Fully Paid’: 76.3% et 95.4% Attention: ces moyennes et medianes varient en fonction de l’echantillon aleatoire Conclusion: il faut adapter le seuil d’allocation. Un seuil de 50% prevoirait trop de defaut!

# ‘class’ contient l’allocation standard des previsions par modalite au seuil de probabilite de base de 50%

qda.class <- qda.pred$class  
qda.class[1:20] # jetez un oeil aux previsions au seuil standard de 50%

## [1] Charged Off Fully Paid Charged Off Fully Paid Fully Paid   
## [6] Fully Paid Fully Paid Fully Paid Charged Off Charged Off  
## [11] Fully Paid Fully Paid Fully Paid Fully Paid Fully Paid   
## [16] Charged Off Fully Paid Fully Paid Fully Paid Fully Paid   
## Levels: Charged Off Fully Paid

Il y a tres peu de previsions de ‘Charged Off’ au seuil de 50% de probabilite

# Creation d’une matrice de confusion: previsions vs. realisations

table(qda.class, dta\_m[-train,]$loan\_status)

##   
## qda.class Charged Off Fully Paid  
## Charged Off 3706 7736  
## Fully Paid 5236 33840

Et sa version etendue avec les % en utilisant Table.Ratio() creee auparavant

Table.Ratio(qda.class, dta\_m[-train,]$loan\_status)

## Charged Off Fully Paid row\_ratio  
## Charged Off 3706.00 7736.00 0.32  
## Fully Paid 5236.00 33840.00 0.87  
## col\_ratio 0.41 0.81 0.74

Cela peut paraitre eleve au global (~75%), mais ce resultat est decevant: - seuls 36% des ‘Charged Off’ sont predit ‘Charge Off’ par le modele - seuls 32% des predictions de ‘Charge Off’ sont effectivement ‘Charged Off’ Attention: ces resultats peuvent varier en fonction des echantillons aleatoires generes

# Utilisation de la fonction Confusion.Table deja creee: elle ajuste le seuil de probabilite et genere une matrice de confusion

Notez: la QDA est deja realisee. Il s’agit de moduler le seuil d’allocation des previsions

Mise en oeuvre de la fonction : etape par etape (indirecte): on prend soin de definir les parametre un a un

actual <- dta\_m[-train, "loan\_status"] # actual status over the 'train' subset  
pred <- qda.pred$posterior[,1] # predicted probabilities (1st column = 'Failed') from QDA model  
proba.threshold <- mean(pred) # average predicted probability instead of 50% probability threshold  
Confusion.Table(actual, pred, proba.threshold)

## Actual.Failed Actual.Valid Accuracy  
## Pred.Failed 4763.0000000 1.100000e+04 0.3021633  
## Pred.Valid 4179.0000000 3.057600e+04 0.8797583  
## Accuracy 0.5326549 7.354243e-01 0.6995328

round(Confusion.Table(actual, pred, proba.threshold),2)

## Actual.Failed Actual.Valid Accuracy  
## Pred.Failed 4763.00 11000.00 0.30  
## Pred.Valid 4179.00 30576.00 0.88  
## Accuracy 0.53 0.74 0.70

Mise ne oeuvre directe: les parametres sont definis a l’interieur au risque d’etre moins visible

round(Confusion.Table(dta\_m[-train, "loan\_status"], qda.pred$posterior[,1], median(qda.pred$posterior[,1])),2)

## Actual.Failed Actual.Valid Accuracy  
## Pred.Failed 6512.00 18747.00 0.26  
## Pred.Valid 2430.00 22829.00 0.90  
## Accuracy 0.73 0.55 0.58

Utiliser la moyenne ou la mediane comme seuil de probabilite pour distinguer les previsions re-alloue les taux de succes: - le taux de succes global diminue - le taux d’identification des defauts effectifs augmente - et la qualite des previsions de defaut decroit

# Creation d’une fonction Run.n.random.qdareg

* Pour realiser ‘n’ regressions lineaires discriminantes
* Sur le dataset ‘dta\_m’
* Et sur le subset in-sample ‘train’ de proportion ‘insample\_prop’
* Et, qui renvoie la pertinence globale de la matrice de confusion des previsions

Run.n.random.qdareg <- function(n = 2, dta, insample\_prop = 0.8, threshold = c("mean", "median", "50%")) {  
 res <- matrix(, nrow = n, ncol = 5)  
 for (i in 1:n) {  
 train <- sample(dim(dta)[1], size = insample\_prop \* dim(dta)[1], replace = FALSE)  
 qda.fit <- qda(loan\_status ~ . , data = dta, subset = train)  
 qda.pred <- predict(qda.fit, dta[-train,])  
 qda.class <- qda.pred$class  
   
 actual <- dta\_m[-train, "loan\_status"] # actual status over the 'train' subset  
 pred <- qda.pred$posterior[,1] # predicted probabilities (1st column = 'Failed') from LDA model  
 proba.threshold <- switch(threshold,  
 "mean" = mean(pred), # average predicted probability  
 "median" = median(pred), # median predicted probability  
 "50%" = 0.5) # 50% probability  
 Confusion.Table(actual, pred, proba.threshold)  
 res[i,1] <- round(Confusion.Table(actual, pred, proba.threshold)[3,3],2)  
 res[i,2] <- round(Confusion.Table(actual, pred, proba.threshold)[3,1],2)  
 res[i,3] <- round(Confusion.Table(actual, pred, proba.threshold)[1,3],2)  
 res[i,4] <- round(Confusion.Table(actual, pred, proba.threshold)[3,2],2)  
 res[i,5] <- round(Confusion.Table(actual, pred, proba.threshold)[2,3],2)  
 colnames(res) <- c("Global", "Actual.Failed", "Pred.Failed", "Actual.Valid", "Pred.Valid")  
   
 }  
 return(res)   
}

# Application: n = 10 regressions quadratiques discriminantes, insample\_prop = 80%, threshold = “mean”

Attention: cela prend un peu de temps avec 10 regressions quadratiques (envisager le calcul parallele!?)

Run.n.random.qdareg(n = 10, dta\_m, insample\_prop = 0.8, threshold = "mean")

## Global Actual.Failed Pred.Failed Actual.Valid Pred.Valid  
## [1,] 0.72 0.48 0.31 0.77 0.87  
## [2,] 0.71 0.49 0.31 0.76 0.87  
## [3,] 0.71 0.48 0.31 0.76 0.87  
## [4,] 0.71 0.50 0.31 0.75 0.87  
## [5,] 0.70 0.50 0.30 0.75 0.87  
## [6,] 0.71 0.50 0.31 0.75 0.87  
## [7,] 0.70 0.51 0.30 0.75 0.87  
## [8,] 0.70 0.53 0.30 0.73 0.88  
## [9,] 0.71 0.48 0.30 0.76 0.87  
## [10,] 0.71 0.50 0.31 0.75 0.87

# Nettoyage des donnees

rm(train, qda.class, qda.pred, qda.fit, dta\_m, pred, actual, proba.threshold)