Reg\_logistic

SOSSOU Gilles

5 février 2019

sqAnoDf <- read\_tsv(file = "anonymized-sq-dataset.tsv")

## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## MEDIA = col\_character(),  
## BARCODE = col\_character(),  
## LABEL = col\_character()  
## )

## See spec(...) for full column specifications.

# set-up ------------------------------------------------------------------  
useSeed <- TRUE  
nSeed <- 123456789  
nCores <- detectCores()  
pSubSample <- 0.2  
indResp <- 1  
  
# list predictive features to be included in training process -------------  
# features are organized by rows of 5  
responsesOfInterest <- c("LABEL")  
featuresOfInterest <- paste0("X", 1:144)  
  
  
# load data set and give generic name -------------------------------------  
if(useSeed)  
 set.seed(nSeed)  
  
compDf <- sqAnoDf %>%  
 dplyr::mutate(LABEL = as.factor(LABEL))%>%  
 dplyr::select(responsesOfInterest[indResp],  
 featuresOfInterest) %>%  
 na.omit(.) %>%  
 dplyr::group\_by\_(responsesOfInterest[indResp]) %>%  
 dplyr::sample\_frac(.2) %>%  
 dplyr::ungroup()  
dim(compDf)

## [1] 1044 145

## Notre jeu de données a été téléchargé et nous aurons à utiliser les 20 % pour le moment afin de tester nos programmes

## Partition des données en données test et training à base des Labels

TrainIndex <- createDataPartition(compDf$LABEL, p = .7,list = FALSE, times = 1)  
  
Train <- compDf[ TrainIndex,]  
Test <- compDf[-TrainIndex,]

## Les données sont partitionnées à 70% pour l’entrainement et 30% pour le test. Ceci en tenant compte aussi de la proportion de répartition des LABELS dans notre jeu de données initiales

# Visualisation de la repartition

round(prop.table(table(Train$LABEL)),2)

##   
## L0 L1 L2 L3 L4   
## 0.36 0.13 0.13 0.13 0.25

round(prop.table(table(compDf$LABEL)),2)

##   
## L0 L1 L2 L3 L4   
## 0.36 0.13 0.13 0.13 0.25

## Effectivement la repartition a été faite dans les mêmes proportions que celle des données totales

# Entrainement du modele sur les données d’entrainement

modele\_reg\_logistic <- train(LABEL ~ .,data = Train,method = "LogitBoost")  
  
print(modele\_reg\_logistic)

## Boosted Logistic Regression   
##   
## 732 samples  
## 144 predictors  
## 5 classes: 'L0', 'L1', 'L2', 'L3', 'L4'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 732, 732, 732, 732, 732, 732, ...   
## Resampling results across tuning parameters:  
##   
## nIter Accuracy Kappa   
## 11 0.8419794 0.7832984  
## 21 0.8326936 0.7718311  
## 31 0.8251820 0.7624839  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was nIter = 11.

LogitBoost.params <- getModelInfo("LogitBoost")  
LogitBoost.params$LogitBoost$parameters

## parameter class label  
## 1 nIter numeric # Boosting Iterations

## La méthode utilisée pour faire notre regression logistique est celle de “LOGITBOOST”. L’accuracy est utilié pour choisir le meilleur modele selon le plus élevé. Et donc ici c’est 84.28% qui est le meilleur avec 11 “iterations”

# Observations de la prédiction sur les données test

predict.reg\_logistic <- predict(modele\_reg\_logistic,Test)  
  
confusionMatrix(predict.reg\_logistic, Test$LABEL)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L0 L1 L2 L3 L4  
## L0 106 2 0 3 1  
## L1 3 32 6 0 0  
## L2 2 3 31 5 0  
## L3 1 0 0 22 4  
## L4 0 0 0 6 69  
##   
## Overall Statistics  
##   
## Accuracy : 0.8784   
## 95% CI : (0.8356, 0.9133)  
## No Information Rate : 0.3784   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8374   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: L0 Class: L1 Class: L2 Class: L3 Class: L4  
## Sensitivity 0.9464 0.8649 0.8378 0.61111 0.9324  
## Specificity 0.9674 0.9653 0.9614 0.98077 0.9730  
## Pos Pred Value 0.9464 0.7805 0.7561 0.81481 0.9200  
## Neg Pred Value 0.9674 0.9804 0.9765 0.94796 0.9774  
## Prevalence 0.3784 0.1250 0.1250 0.12162 0.2500  
## Detection Rate 0.3581 0.1081 0.1047 0.07432 0.2331  
## Detection Prevalence 0.3784 0.1385 0.1385 0.09122 0.2534  
## Balanced Accuracy 0.9569 0.9151 0.8996 0.79594 0.9527

## Sur les données test, on obtient une accuracy de 81.94%. Les classes L0 et L4 les forts taux de sensitivité et de spécificité. Ce qui nous amene à réfléchir sur le fait des proportions des classes, s’il faudrait traiter la prédiction en tenant compte des déséquilibresde proportion des classes

## Toutefois nous allons refaire cette methode en dissociant selon les différents Media, puisque la répartion ne reste pas les mêmes aux seins des médias

# Déclaration des listes

medias <- unique(sqAnoDf$MEDIA)  
df<-list()  
traini<-list()  
testi<-list()  
predict.reg\_logistic<-list()  
Response<-list()  
modele\_reg\_logistic<-list()

# Repartition des données en fonction des MEDIAS

if(useSeed)  
 set.seed(nSeed)  
  
for( i in 1:length(medias)){  
df[[i]] <- sqAnoDf %>%  
 dplyr:: filter(MEDIA==medias[i]) %>%  
 dplyr:: mutate(LABEL = as.factor(LABEL))%>%  
 dplyr:: select(responsesOfInterest,featuresOfInterest)%>%  
 dplyr::sample\_frac(.2) %>%  
 dplyr::ungroup()  
   
}  
dim(df[[4]])

## [1] 544 145

# Séparation des données en training et test

for( i in 1:length(medias)){  
traini[[i]] <- df[[i]]%>% sample\_frac(0.7)   
testi[[i]]<- anti\_join(df[[i]], traini[[i]])  
}

prop.table(table(df[[4]]$LABEL))

##   
## L0 L1 L2 L3 L4   
## 0.3014706 0.1415441 0.1488971 0.1433824 0.2647059

# Apprentissage du modèle avec la methode “LogitBoost” de la regression logistique

for( i in 1:length(medias)){  
modele\_reg\_logistic[[i]] <- train(LABEL ~ .,data = traini[[i]],  
 method = "LogitBoost")  
}

## Ici le modèle a été entrainé selon les différentes médias, contrairement au premier cas où il a été entrainé sur tous médias réunis

# Application de la modelisation sur les données test

for( i in 1:length(medias)){  
predict.reg\_logistic[[i]] <- predict(modele\_reg\_logistic[[i]],testi[[i]])  
Response[[i]] <- confusionMatrix(predict.reg\_logistic[[i]], testi[[i]]$LABEL)  
  
}

# Visualisation des matrices de confusion

Response[[4]]

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L0 L1 L2 L3 L4  
## L0 43 5 0 2 0  
## L1 3 16 2 0 0  
## L2 0 1 17 2 1  
## L3 0 0 2 7 2  
## L4 0 2 0 1 37  
##   
## Overall Statistics  
##   
## Accuracy : 0.8392   
## 95% CI : (0.7685, 0.8952)  
## No Information Rate : 0.3217   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7874   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: L0 Class: L1 Class: L2 Class: L3 Class: L4  
## Sensitivity 0.9348 0.6667 0.8095 0.58333 0.9250  
## Specificity 0.9278 0.9580 0.9672 0.96947 0.9709  
## Pos Pred Value 0.8600 0.7619 0.8095 0.63636 0.9250  
## Neg Pred Value 0.9677 0.9344 0.9672 0.96212 0.9709  
## Prevalence 0.3217 0.1678 0.1469 0.08392 0.2797  
## Detection Rate 0.3007 0.1119 0.1189 0.04895 0.2587  
## Detection Prevalence 0.3497 0.1469 0.1469 0.07692 0.2797  
## Balanced Accuracy 0.9313 0.8123 0.8884 0.77640 0.9479

Response[[3]]

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L0 L1 L2 L3 L4  
## L0 29 1 0 0 0  
## L1 0 1 3 0 0  
## L2 0 0 0 0 0  
## L3 1 0 0 1 0  
## L4 0 0 0 1 12  
##   
## Overall Statistics  
##   
## Accuracy : 0.8776   
## 95% CI : (0.7523, 0.9537)  
## No Information Rate : 0.6122   
## P-Value [Acc > NIR] : 4.118e-05   
##   
## Kappa : 0.7794   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: L0 Class: L1 Class: L2 Class: L3 Class: L4  
## Sensitivity 0.9667 0.50000 0.00000 0.50000 1.0000  
## Specificity 0.9474 0.93617 1.00000 0.97872 0.9730  
## Pos Pred Value 0.9667 0.25000 NaN 0.50000 0.9231  
## Neg Pred Value 0.9474 0.97778 0.93878 0.97872 1.0000  
## Prevalence 0.6122 0.04082 0.06122 0.04082 0.2449  
## Detection Rate 0.5918 0.02041 0.00000 0.02041 0.2449  
## Detection Prevalence 0.6122 0.08163 0.00000 0.04082 0.2653  
## Balanced Accuracy 0.9570 0.71809 0.50000 0.73936 0.9865

Response[[2]]

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L0 L1 L2 L3 L4  
## L0 7 1 0 0 0  
## L1 0 0 1 1 0  
## L2 1 0 3 0 0  
## L3 0 0 3 1 0  
## L4 0 0 0 3 10  
##   
## Overall Statistics  
##   
## Accuracy : 0.6774   
## 95% CI : (0.4863, 0.8332)  
## No Information Rate : 0.3226   
## P-Value [Acc > NIR] : 5.479e-05   
##   
## Kappa : 0.5676   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: L0 Class: L1 Class: L2 Class: L3 Class: L4  
## Sensitivity 0.8750 0.00000 0.42857 0.20000 1.0000  
## Specificity 0.9565 0.93333 0.95833 0.88462 0.8571  
## Pos Pred Value 0.8750 0.00000 0.75000 0.25000 0.7692  
## Neg Pred Value 0.9565 0.96552 0.85185 0.85185 1.0000  
## Prevalence 0.2581 0.03226 0.22581 0.16129 0.3226  
## Detection Rate 0.2258 0.00000 0.09677 0.03226 0.3226  
## Detection Prevalence 0.2581 0.06452 0.12903 0.12903 0.4194  
## Balanced Accuracy 0.9158 0.46667 0.69345 0.54231 0.9286

Response[[1]]

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L0 L1 L2 L3 L4  
## L0 14 2 1 1 0  
## L1 0 2 0 1 0  
## L2 0 1 7 2 0  
## L3 0 0 1 5 3  
## L4 0 0 0 4 13  
##   
## Overall Statistics  
##   
## Accuracy : 0.7193   
## 95% CI : (0.5846, 0.8303)  
## No Information Rate : 0.2807   
## P-Value [Acc > NIR] : 8.32e-12   
##   
## Kappa : 0.6356   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: L0 Class: L1 Class: L2 Class: L3 Class: L4  
## Sensitivity 1.0000 0.40000 0.7778 0.38462 0.8125  
## Specificity 0.9070 0.98077 0.9375 0.90909 0.9024  
## Pos Pred Value 0.7778 0.66667 0.7000 0.55556 0.7647  
## Neg Pred Value 1.0000 0.94444 0.9574 0.83333 0.9250  
## Prevalence 0.2456 0.08772 0.1579 0.22807 0.2807  
## Detection Rate 0.2456 0.03509 0.1228 0.08772 0.2281  
## Detection Prevalence 0.3158 0.05263 0.1754 0.15789 0.2982  
## Balanced Accuracy 0.9535 0.69038 0.8576 0.64685 0.8575

tab=cbind("observé"=testi[[4]]$LABEL,"predite"=predict.reg\_logistic[[4]])  
head(tab)

## observé predite  
## [1,] 3 4  
## [2,] 1 NA  
## [3,] 5 5  
## [4,] 3 NA  
## [5,] 3 3  
## [6,] 1 1

## Nous pouvons observer que certaines observations n’ont pas pu obtenir des réponses de prédictions, ce qui nous donne des NA sur leur valeur de predictions. Le plus bas accuracy est obtenue dans le M1

# Selection des Variables importantes

# Boruta ------------------------------------------------------------------  
#resBoruta <- Boruta(x = compDf[, featuresOfInterest],  
# y = compDf[, responsesOfInterest[indResp]][[1]],  
# doTrace = 2,  
# num.threads = nCores - 1,  
# ntree = 100)

# Nous obtenons 82 variables confirmées

# Importation des variables importantes

load("dataImpVar.Rdata")  
dataImpVar

# Déclaration des listes

medias <- unique(sqAnoDf$MEDIA)  
  
df2<-list()  
trainj<-list()  
testj<-list()  
predict.reg\_logistic\_Imp<-list()  
Response\_Imp<-list()  
modele\_reg\_logistic\_Imp<-list()

# Repartition des données en fonction des MEDIAS

if(useSeed)  
 set.seed(nSeed)  
  
ImpVar<-names(dataImpVar)  
  
for( j in 1:length(medias)){  
df2[[j]] <- sqAnoDf %>%  
 dplyr:: filter(MEDIA==medias[j]) %>%  
 dplyr:: mutate(LABEL = as.factor(LABEL))%>%  
 dplyr:: select(ImpVar)%>%  
 dplyr:: sample\_frac(.2) %>%  
 dplyr:: ungroup()  
   
}  
  
dim(df2[[1]])

## [1] 221 83

dim(df2[[2]])

## [1] 110 83

dim(df2[[3]])

## [1] 168 83

dim(df2[[4]])

## [1] 544 83

# Separation des donnees selon les differents medias

for( j in 1:length(medias)){  
trainj[[j]] <- df2[[j]]%>% sample\_frac(0.7)   
testj[[j]]<- anti\_join(df2[[j]], traini[[j]])  
}

(table(df2[[1]]$LABEL))

##   
## L0 L1 L2 L3 L4   
## 70 30 35 37 49

(table(df2[[2]]$LABEL))

##   
## L0 L1 L2 L3 L4   
## 29 16 15 16 34

(table(df2[[3]]$LABEL))

##   
## L0 L1 L2 L3 L4   
## 100 14 9 11 34

(table(df2[[4]]$LABEL))

##   
## L0 L1 L2 L3 L4   
## 164 77 81 78 144

## Nous observons bien la repartition des labels en fonction des medias

# Apprentissage du modèle avec la methode “LogitBoost” de la regression logistique

for( j in 1:length(medias)){  
modele\_reg\_logistic\_Imp[[j]] <- train(LABEL ~ .,data = trainj[[j]],  
 method = "LogitBoost")  
}

# Application de la modelisation sur les données test

for( j in 1:length(medias)){  
predict.reg\_logistic\_Imp[[j]] <- predict(modele\_reg\_logistic\_Imp[[j]],testj[[j]])  
Response\_Imp[[j]] <- confusionMatrix(predict.reg\_logistic\_Imp[[j]], testj[[j]]$LABEL)  
  
}

# Visualisation des matrices de confusion

Response\_Imp[[1]]

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L0 L1 L2 L3 L4  
## L0 17 3 1 1 0  
## L1 0 2 0 0 0  
## L2 0 1 9 2 0  
## L3 0 0 1 5 1  
## L4 0 0 0 3 14  
##   
## Overall Statistics  
##   
## Accuracy : 0.7833   
## 95% CI : (0.658, 0.8793)  
## No Information Rate : 0.2833   
## P-Value [Acc > NIR] : 1.377e-15   
##   
## Kappa : 0.7164   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: L0 Class: L1 Class: L2 Class: L3 Class: L4  
## Sensitivity 1.0000 0.33333 0.8182 0.45455 0.9333  
## Specificity 0.8837 1.00000 0.9388 0.95918 0.9333  
## Pos Pred Value 0.7727 1.00000 0.7500 0.71429 0.8235  
## Neg Pred Value 1.0000 0.93103 0.9583 0.88679 0.9767  
## Prevalence 0.2833 0.10000 0.1833 0.18333 0.2500  
## Detection Rate 0.2833 0.03333 0.1500 0.08333 0.2333  
## Detection Prevalence 0.3667 0.03333 0.2000 0.11667 0.2833  
## Balanced Accuracy 0.9419 0.66667 0.8785 0.70686 0.9333

Response\_Imp[[2]]

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L0 L1 L2 L3 L4  
## L0 8 1 0 0 0  
## L1 0 0 1 2 0  
## L2 0 0 2 0 0  
## L3 0 0 3 0 0  
## L4 0 0 0 3 10  
##   
## Overall Statistics  
##   
## Accuracy : 0.6667   
## 95% CI : (0.4719, 0.8271)  
## No Information Rate : 0.3333   
## P-Value [Acc > NIR] : 0.0001938   
##   
## Kappa : 0.5509   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: L0 Class: L1 Class: L2 Class: L3 Class: L4  
## Sensitivity 1.0000 0.00000 0.33333 0.0000 1.0000  
## Specificity 0.9545 0.89655 1.00000 0.8800 0.8500  
## Pos Pred Value 0.8889 0.00000 1.00000 0.0000 0.7692  
## Neg Pred Value 1.0000 0.96296 0.85714 0.8148 1.0000  
## Prevalence 0.2667 0.03333 0.20000 0.1667 0.3333  
## Detection Rate 0.2667 0.00000 0.06667 0.0000 0.3333  
## Detection Prevalence 0.3000 0.10000 0.06667 0.1000 0.4333  
## Balanced Accuracy 0.9773 0.44828 0.66667 0.4400 0.9250

Response\_Imp[[3]]

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L0 L1 L2 L3 L4  
## L0 29 1 0 1 0  
## L1 0 1 3 0 0  
## L2 0 0 0 0 0  
## L3 1 0 0 1 0  
## L4 0 0 0 1 12  
##   
## Overall Statistics  
##   
## Accuracy : 0.86   
## 95% CI : (0.7326, 0.9418)  
## No Information Rate : 0.6   
## P-Value [Acc > NIR] : 6.12e-05   
##   
## Kappa : 0.75   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: L0 Class: L1 Class: L2 Class: L3 Class: L4  
## Sensitivity 0.9667 0.5000 0.00 0.3333 1.0000  
## Specificity 0.9000 0.9375 1.00 0.9787 0.9737  
## Pos Pred Value 0.9355 0.2500 NaN 0.5000 0.9231  
## Neg Pred Value 0.9474 0.9783 0.94 0.9583 1.0000  
## Prevalence 0.6000 0.0400 0.06 0.0600 0.2400  
## Detection Rate 0.5800 0.0200 0.00 0.0200 0.2400  
## Detection Prevalence 0.6200 0.0800 0.00 0.0400 0.2600  
## Balanced Accuracy 0.9333 0.7188 0.50 0.6560 0.9868

Response\_Imp[[4]]

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L0 L1 L2 L3 L4  
## L0 43 2 1 1 0  
## L1 3 15 2 1 0  
## L2 0 1 15 1 1  
## L3 0 0 1 8 2  
## L4 1 2 0 3 36  
##   
## Overall Statistics  
##   
## Accuracy : 0.8417   
## 95% CI : (0.7702, 0.8981)  
## No Information Rate : 0.3381   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7899   
## Mcnemar's Test P-Value : 0.6549   
##   
## Statistics by Class:  
##   
## Class: L0 Class: L1 Class: L2 Class: L3 Class: L4  
## Sensitivity 0.9149 0.7500 0.7895 0.57143 0.9231  
## Specificity 0.9565 0.9496 0.9750 0.97600 0.9400  
## Pos Pred Value 0.9149 0.7143 0.8333 0.72727 0.8571  
## Neg Pred Value 0.9565 0.9576 0.9669 0.95312 0.9691  
## Prevalence 0.3381 0.1439 0.1367 0.10072 0.2806  
## Detection Rate 0.3094 0.1079 0.1079 0.05755 0.2590  
## Detection Prevalence 0.3381 0.1511 0.1295 0.07914 0.3022  
## Balanced Accuracy 0.9357 0.8498 0.8822 0.77371 0.9315

#tab=cbind("observé"=testi[[3]]$LABEL,"predite"=predict.reg\_logistic[[3]])

## En visualisant la matrice de confusion selon les différents media,nous retenons que l’accuracy varie selon chaque media, ce qui serait du au contenu des données.

## Le plus bas accuracy est celui de la media M1 ou on observe 66%. Tout comme dans le cas des differents medias avec toutes les variables. La meilleure prediction est obtenue dans le cas du M2 avec 87% tout comme ca été aussi le meilleur dans le cas de toutes les variables incluses. Nous allons plus tard chercher à savoir si le cas du M1 est du à l’effectif le plus bas don’t il dispose? Serions nous dans experimentation où nous manquions de données?