Packages utilisés

```
library(unbalanced)
```

```
## Le chargement a nécessité le package : mlr
```

```
## Le chargement a nécessité le package : ParamHelpers
```

```
## Warning message: 'mlr' is in 'maintenance-only' mode since July 2019.
## Future development will only happen in 'mlr3'
## (<https://mlr3.mlr-org.com>). Due to the focus on 'mlr3' there might be
## uncaught bugs meanwhile in {mlr} - please consider switching.
```

```
## Le chargement a nécessité le package : foreach
```

```
## Le chargement a nécessité le package : doParallel
```

```
## Le chargement a nécessité le package : iterators
```

```
## Le chargement a nécessité le package : parallel
```

```
library(caTools)
library(lubridate)
```

```
##
## Attachement du package : 'lubridate'
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attachement du package : 'pROC'
```

```
## Les objets suivants sont masqués depuis 'package:stats':
##
##     cov, smooth, var
```

```
library(LaplacesDemon)
```

```
##
## Attachement du package : 'LaplacesDemon'
```

```
## Les objets suivants sont masqués depuis 'package:lubridate':
##
##     dst, interval
```

```
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'
```

```
## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(randomForest)
```

```
## randomForest 4.7-1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attachement du package : 'randomForest'
## L'objet suivant est masqué depuis 'package:dplyr':
##
##     combine
```

```
library(keras)
library(tensorflow)
```

```
##
## Attachement du package : 'tensorflow'
## L'objet suivant est masqué depuis 'package:mlr':
##
##     train
```

```
library(BBmisc)
```

```
##
## Attachement du package : 'BBmisc'
## L'objet suivant est masqué depuis 'package:keras':
##
##     normalize
## Les objets suivants sont masqués depuis 'package:dplyr':
##
##     coalesce, collapse
## L'objet suivant est masqué depuis 'package:base':
##
##     isFALSE
```

PRETRAITEMENT

1) Importation des données et définition des variables

```
remove(list=objects())
data=read.csv2('octroi_RCI.csv',header=TRUE)

duree=data$DUREE_CONTRAT
anc_emploi=data$ANC_EMPLOI
sit_fam=data$SITUATION_FAM
loge=data$MODE_LOGT
age_ve=data$AGE_VEH
NO=data$VN_VO
marque=data$MARQUE
prix_ve=data$PRIX_VEH
apport=data$MT_APPORT
finance=data$MT_FINANCE
mens=data$MT_MENS
vr_ball=data$VR_BALLON
prest=data$MT_PREST
```

```
assur=data$MT_ASSUR
age_cli=data$age_cli
ancien_rci=data$anciennete_rci
pc_appo=data$pc_appo
def=data$def12_31
```

2) Test des NA's

```
n = nrow(data)
na_age_cli = 0
na_age_ve = 0
na_anc_emploi = 0
na_pc_appo = 0
na_apport = 0
na_assur = 0
na_duree = 0
na_finance = 0
na_loge = 0
na_marque = 0
na_mens = 0
na_no = 0
na_prest = 0
na_prix_ve = 0
na_sit_fam = 0
na_vr_ball = 0
na_ancien_rci = 0
na_def=0
for (i in 1:n){
  if (is.na(age_cli[i])){
    na_age_cli = na_age_cli+1
  }
  if (is.na(age_ve[i])){
    na_age_ve = na_age_ve+1
    }
  if (is.na(anc_emploi[i])){
    na_anc_emploi = na_anc_emploi+1
    }
  if (is.na(ancien_rci[i])){
    na_ancien_rci = na_ancien_rci+1
    }
  if (is.na(pc_appo[i])){
    na_pc_appo = na_pc_appo+1
  }
  if (is.na(apport[i])){
    na_apport = na_apport+1
  }
  if (is.na(assur[i])){
    na_assur = na_assur+1
  }
  if (is.na(duree[i])){
    na_duree = na_duree+1
  }
  if (is.na(finance[i])){
    na_finance = na_finance+1
```

```
  }
  if (is.na(loge[i])){
    na_loge = na_loge+1
  }
  if (is.na(marque[i])){
    na_marque = na_marque+1
  }
  if (is.na(mens[i])){
    na_mens = na_mens+1
  }
  if (is.na(NO[i])){
    na_no = na_no+1
  }
  if (is.na(prest[i])){
    na_prest = na_prest+1
  }
  if (is.na(prix_ve[i])){
    na_prix_ve = na_prix_ve+1
  }
  if (is.na(sit_fam[i])){
    na_sit_fam = na_sit_fam+1
  }
  if (is.na(vr_ball[i])){
    na_vr_ball = na_vr_ball+1
  }
  if (is.na(def[i])){
    na_def = na_def+1
  }
}
```

On observe que les variables qui ont des NA's sont : ancien_rci(7271) assur(6755) age_ve(6003) loge(108) prest(1280) vr_ball(5238)

3) Remplacement des NA's

```
n = nrow(data)


for (i in 1:n){
  if(is.na(assur[i])){
    assur[i]=0
  }
}

for (i in 1:n){
  if(is.na(prest[i])){
    prest[i]=0
  }
}

for (i in 1:n){
  if(is.na(loge[i])){
    loge[i]=3
  }
}
```

```
for (i in 1:n){
  if(is.na(age_ve[i])){
    age_ve[i]=0
  }
}

for (i in 1:n){
  if(is.na(vr_ball[i])){
    vr_ball[i]=0
  }
}

for (i in 1:n){
  if(is.na(ancien_rci[i])){
    ancien_rci[i]=-1
  }
}
```

Les NA's des variables prest, assur, age_ve et vr_ball sont remplacés par 0. Les variables loge et ancien_RCI prennent une nouvelle modalité pour les NA's. La variable ancien_RCI sera retirée de l'analyse.

4) Création de Features

```
x=rep(1,nrow(data))


# ANC_EMPLOI
segments_emploi<-function(x){
  for (i in 1:nrow(data) ) {
    if ( x[i] <= 104 ) {x[i] = 1 }
    else { x[i] = 0 }
  }
  return(x)
}
flag_emploi_inf_104 = segments_emploi(anc_emploi)

segments_emploi<-function(x){
  for (i in 1:nrow(data) ) {
    if ( x[i] > 104 & x[i] <= 260 ) {x[i] = 1 }
    else { x[i] = 0 }
  }
  return(x)
}
flag_emploi_104_260 = segments_emploi(anc_emploi)

segments_emploi<-function(x){
  for (i in 1:nrow(data) ) {
    if ( x[i] > 260 & x[i] <= 520 ) {x[i] = 1 }
    else { x[i] = 0 }
  }
  return(x)
}
flag_emploi_260_520 = segments_emploi(anc_emploi)
```

```r
segments_emploi<-function(x){
  for (i in 1:nrow(data) ) {
    if ( x[i] > 520 ) {x[i] = 1 }
    else { x[i] = 0 }
  }
  return(x)
}
flag_emploi_sup_520 = segments_emploi(anc_emploi)


flag_emploi = cbind( flag_emploi_inf_104, flag_emploi_104_260, flag_emploi_260_520, flag_emploi_sup_520



# AGE_CLI
segments_age<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] <= 25 ){ x[i] = 1 }
    else { x[i] =0 }
  }
  return (x)
}
flag_agecli_inf_25 =segments_age(age_cli)

segments_age<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 25 & x[i] <= 40){ x[i] = 1 }
    else { x[i] =0 }
  }
  return (x)
}
flag_agecli_25_40 =segments_age(age_cli)

segments_age<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 40 & x[i] <= 60){ x[i] = 1 }
    else { x[i] =0 }
  }
  return (x)
}
flag_agecli_40_60 =segments_age(age_cli)

segments_age<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 60 ){ x[i] = 1 }
    else { x[i] =0 }
  }
  return (x)
}
flag_agecli_sup_60 =segments_age(age_cli)


flag_agecli = cbind( flag_agecli_inf_25, flag_agecli_25_40, flag_agecli_40_60, flag_agecli_sup_60  )
```

```
#AGE_VEH
segments_age_ve<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == 0){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ageve_neuf =segments_age_ve(age_ve)

segments_age_ve<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 0 & x[i] <= 25 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ageve_0_25 =segments_age_ve(age_ve)

segments_age_ve<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 25 & x[i] <= 50 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ageve_25_50 =segments_age_ve(age_ve)

segments_age_ve<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 50 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ageve_sup_50 =segments_age_ve(age_ve)


flag_ageve = cbind( flag_ageve_neuf, flag_ageve_0_25, flag_ageve_25_50, flag_ageve_sup_50 )


# PRIX_VEH
segments_prix_ve<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] <= 10000 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prix_ve_inf_10k =segments_prix_ve(prix_ve)

segments_prix_ve<-function(x){
  for (i in 1:nrow(data)){
```

```
    if ( x[i] > 10000 & x[i] <= 15000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prix_ve_10k_15k =segments_prix_ve(prix_ve)

segments_prix_ve<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 15000 & x[i] <= 20000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prix_ve_15k_20k =segments_prix_ve(prix_ve)

segments_prix_ve<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 20000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prix_ve_sup_20k =segments_prix_ve(prix_ve)


flag_prix_ve = cbind( flag_prix_ve_inf_10k, flag_prix_ve_10k_15k, flag_prix_ve_15k_20k, flag_prix_ve_su



# PC_MENS
pc_mens=(mens/prix_ve)*100
segments_pc_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 1.2581){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_mens_inf_1.2581 = segments_pc_mens(pc_mens)

segments_pc_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 1.2581 & x[i] <= 1.5149){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_mens_1.2581_1.5149 = segments_pc_mens(pc_mens) #1.51=1er quantile

segments_pc_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 1.5149 & x[i] <= 1.7745 ){ x[i] = 1 } # moyenne 1.7745
```

```
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_mens_1.5149_1.7745= segments_pc_mens(pc_mens)

segments_pc_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 1.7745 ){ x[i] = 1 } # moyenne 1.7745
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_mens_sup_1.7745= segments_pc_mens(pc_mens)


flag_pc_mens = cbind( flag_pc_mens_inf_1.2581,flag_pc_mens_1.2581_1.5149,flag_pc_mens_1.5149_1.7745, fla




#ASSUR
segments_noassur<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == 0  ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_noassur = segments_noassur(assur)


segments_assur<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] >0 & x[i] <= 500 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_assur_inf_500 = segments_assur(assur)

segments_assur<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 500 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_assur_sup_500 = segments_assur(assur)

flag_assur = cbind( flag_noassur, flag_assur_inf_500, flag_assur_sup_500)
```

```r
# VR_BALL

segments_ball<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==0 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_noball = segments_ball(vr_ball)

segments_ball<-function(x){       #summary(vr_ball)  moy= 9390
  for (i in 1:nrow(data)){
    if (x[i] >0  & x[i] <= 10000 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ball_inf_10000 = segments_ball(vr_ball)

segments_ball<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 10000 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ball_sup_10000 = segments_ball(vr_ball)


flag_ball = cbind( flag_noball, flag_ball_inf_10000, flag_ball_sup_10000 )


# MT_APPORT

segments_apport<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 5000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_apport_inf_5000 = segments_apport(apport)


segments_apport<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 5000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_apport_sup_5000 = segments_apport(apport)
```

```r
flag_apport = cbind( flag_apport_inf_5000, flag_apport_sup_5000 )




# MARQUE
segments_marque<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == "REN") { x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_marque_RENAULT = as.numeric(segments_marque(marque))

segments_marque<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == "DAC"){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_marque_DACIA = as.numeric(segments_marque(marque))

segments_marque<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == "NIS"){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_marque_NISSAN = as.numeric(segments_marque(marque))

segments_marque<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] != "REN" & x[i] != "DAC" & x[i] != "NIS"){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_marque_AUTRE = as.numeric(segments_marque(marque))

flag_marque = cbind( flag_marque_RENAULT,flag_marque_DACIA, flag_marque_NISSAN, flag_marque_AUTRE )


#DUREE_CONTRAT
segments_duree<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 24){ x[i] = 1 }
    else { x[i] = 0 }
  }
```

```
  return (x)
}
flag_duree_inf_24 = segments_duree(duree)

segments_duree<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 24 & x[i] <= 36){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_duree_24_36 = segments_duree(duree)

segments_duree<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 36){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_duree_sup_36 = segments_duree(duree)


flag_duree = cbind( flag_duree_inf_24, flag_duree_24_36, flag_duree_sup_36 )



#PREST

segments_noprest<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==0){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_noprest = segments_noprest(prest)

segments_prest<-function(x){
  for (i in 1:nrow(data)){
    if (x[i]>0 & x[i] <= 140 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}

flag_prest_inf_140 = segments_prest(prest)

segments_prest<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 140 & x[i] <= 400 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
```

```
  return (x)
}
flag_prest_140_400 = segments_prest(prest)

segments_prest<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 400 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prest_sup_400 = segments_prest(prest)


flag_prest = cbind( flag_prest_inf_140, flag_prest_140_400, flag_prest_sup_400, flag_noprest  )



#MODE_LOGT
segments_loge<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==  1 | x[i]==3 | x[i]==4){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_loge_nonprop = segments_loge(loge)

segments_loge<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==  2 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_loge_proprietaire = segments_loge(loge)

flag_loge = cbind(flag_loge_proprietaire, flag_loge_nonprop)



#PC_APPO
segments_pc_appo<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 13){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_appo_inf_13 = segments_pc_appo(pc_appo)

segments_pc_appo<-function(x){
  for (i in 1:nrow(data)){
```

```
    if (x[i] > 13 & x[i] <= 27){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_appo_13_27 = segments_pc_appo(pc_appo)

segments_pc_appo<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 27 & x[i] <= 36){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_appo_27_36 = segments_pc_appo(pc_appo)

segments_pc_appo<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 36){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_appo_sup_36 = segments_pc_appo(pc_appo)


flag_pc_appo = cbind( flag_pc_appo_inf_13, flag_pc_appo_13_27, flag_pc_appo_27_36, flag_pc_appo_sup_36 )


#SIT_FAM
segments_sit_fam<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == 1 | x[i]==3){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_sit_fam_marie_div = segments_sit_fam(sit_fam)

segments_sit_fam<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == 2 | x[i] == 4 | x[i] == 5 | x[i] == 11){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_sit_fam_autre = segments_sit_fam(sit_fam)

flag_sit_fam = cbind( flag_sit_fam_marie_div, flag_sit_fam_autre)


#MENS
segments_mens<-function(x){
```

```
  for (i in 1:nrow(data)){
    if (x[i] <= 200){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_mens_inf_200 = segments_mens(mens)

segments_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 200 & x[i] <= 300){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_mens_200_300 = segments_mens(mens)

segments_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 300 & x[i] <= 400){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_mens_300_400 = segments_mens(mens)

segments_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 400){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_mens_sup_400 = segments_mens(mens)


flag_mens = cbind( flag_mens_inf_200, flag_mens_200_300, flag_mens_300_400, flag_mens_sup_400 )

#BENEF

delta<-function(x){
  for (i in 1:nrow(data)){
  x[i]= (duree[i]*mens[i])-finance[i]+vr_ball[i]}
return (x)
}
benef=delta(vr_ball)

segments_benef<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 1000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
```

```
}
flag_benef_inf_1k = segments_benef(benef)

segments_benef<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 1000 & x[i] <= 2000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_benef_1k_2k = segments_benef(benef)

segments_benef<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 2000 & x[i] <= 3000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_benef_2k_3k = segments_benef(benef)

segments_benef<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 3000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_benef_sup_3k = segments_benef(benef)


flag_benef = cbind( flag_benef_inf_1k, flag_benef_1k_2k, flag_benef_2k_3k, flag_benef_sup_3k)


# FINANCE
segments_finance<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 10000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_finance_inf_10k = segments_finance(finance)

segments_finance<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 10000 & x[i] <= 13000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_finance_10k_13k = segments_finance(finance)
```

```r
segments_finance<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 13000 & x[i] <= 16000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_finance_13k_16k = segments_finance(finance)

segments_finance<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 16000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_finance_sup_16k = segments_finance(finance)


flag_finance = cbind( flag_finance_inf_10k, flag_finance_10k_13k, flag_finance_13k_16k, flag_finance_su

# anciennete-rci
segments_anc<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==-1){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_NA_anc = segments_anc(ancien_rci)

segments_anc<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 2){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_anc_inf_2 = segments_anc(ancien_rci)

segments_anc<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 2 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_anc_sup_2 = segments_anc(ancien_rci)

flag_anc = cbind( flag_NA_anc, flag_anc_inf_2, flag_anc_sup_2 )
```

```
#NO
segments_NO<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == "VN"){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_NO = segments_NO(NO)
flag_NO <- as.numeric(flag_NO)



features=cbind(duree, anc_emploi, sit_fam, loge, age_ve,prix_ve, apport, finance,  mens, vr_ball, prest
data2=as.data.frame(cbind(def,features))


flag=cbind(flag_emploi, flag_agecli, flag_ageve, flag_prix_ve, flag_pc_mens, flag_assur, flag_ball, flag
data_flag=as.data.frame(cbind(def,flag))



data_total <-as.data.frame(cbind(data2,data_flag[,-1]))

write.csv(data_total,"data.csv", row.names = FALSE)
```

SPLIT ET REECHANTILLONNAGE

```
remove(list=objects())
data=read.csv2('data.csv', sep=",",header=TRUE)

data[,1]=as.numeric(data[,1])
data[,2]=as.numeric(data[,2])
data[,3]=as.numeric(data[,3])
data[,4]=as.numeric(data[,4])
data[,5]=as.numeric(data[,5])
data[,6]=as.numeric(data[,6])
data[,7]=as.numeric(data[,7])
data[,8]=as.numeric(data[,8])
data[,9]=as.numeric(data[,9])
data[,10]=as.numeric(data[,10])
data[,11]=as.numeric(data[,11])
data[,12]=as.numeric(data[,12])
data[,13]=as.numeric(data[,13])
data[,14]=as.numeric(data[,14])
data[,15]=as.numeric(data[,15])
data[,16]=as.numeric(data[,16])
data[,17]=as.numeric(data[,17])
data[,18]=as.numeric(data[,18])
```

SPLIT

```
set.seed(28100)
split<-sample.split(data[,1],SplitRatio = 0.8)
train <- subset(data,split==TRUE)
test <- subset(data,split==FALSE)
write.csv(train,"train_non_reech.csv", row.names = FALSE)
write.csv(test,"test.csv", row.names = FALSE)
```

REECHANTILLONAGE DE TRAIN

```
train <- train[,1:18]
def_fact <- as.factor(train$def)
smote <- ubSMOTE(train[,-1], def_fact, perc.over = 2000, k = 4, perc.under = 400, verbose = TRUE)
summary(smote$Y)
```

```
##    0    1
## 6240 1638
```

```
summary(def_fact)
```

```
##    0    1
## 6687   78
```

```
def <- as.numeric(smote$Y)
data <- as.data.frame(cbind(def, smote$X))
```

CALCUL DES FLAG DE TRAIN

```
duree=data$duree
anc_emploi=data$anc_emploi
sit_fam=data$sit_fam
loge=data$loge
age_ve=data$age_ve
prix_ve=data$prix_ve
apport=data$apport
finance=data$finance
mens=data$mens
vr_ball=data$vr_ball
prest=data$prest
assur=data$assur
age_cli=data$age_cli
ancien_rci=data$ancien_rci
pc_appo=data$pc_appo
def=data$def



x=rep(1,nrow(data))


# ANC_EMPLOI
segments_emploi<-function(x){
  for (i in 1:nrow(data) ) {
    if ( x[i] <= 104 ) {x[i] = 1 }
    else { x[i] = 0 }
  }
  return(x)
```

```
}
flag_emploi_inf_104 = segments_emploi(anc_emploi)

segments_emploi<-function(x){
  for (i in 1:nrow(data) ) {
    if ( x[i] > 104 & x[i] <= 260 ) {x[i] = 1 }
    else { x[i] = 0 }
  }
  return(x)
}
flag_emploi_104_260 = segments_emploi(anc_emploi)

segments_emploi<-function(x){
  for (i in 1:nrow(data) ) {
    if ( x[i] > 260 & x[i] <= 520 ) {x[i] = 1 }
    else { x[i] = 0 }
  }
  return(x)
}
flag_emploi_260_520 = segments_emploi(anc_emploi)

segments_emploi<-function(x){
  for (i in 1:nrow(data) ) {
    if ( x[i] > 520 ) {x[i] = 1 }
    else { x[i] = 0 }
  }
  return(x)
}
flag_emploi_sup_520 = segments_emploi(anc_emploi)


flag_emploi = cbind( flag_emploi_inf_104, flag_emploi_104_260, flag_emploi_260_520, flag_emploi_sup_520



# AGE_CLI
segments_age<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] <= 25 ){ x[i] = 1 }
    else { x[i] =0 }
  }
  return (x)
}
flag_agecli_inf_25 =segments_age(age_cli)

segments_age<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 25 & x[i] <= 40){ x[i] = 1 }
    else { x[i] =0 }
  }
  return (x)
}
flag_agecli_25_40 =segments_age(age_cli)
```

```
segments_age<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 40 & x[i] <= 60){ x[i] = 1 }
    else { x[i] =0 }
  }
  return (x)
}
flag_agecli_40_60 =segments_age(age_cli)

segments_age<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 60 ){ x[i] = 1 }
    else { x[i] =0 }
  }
  return (x)
}
flag_agecli_sup_60 =segments_age(age_cli)


flag_agecli = cbind( flag_agecli_inf_25, flag_agecli_25_40, flag_agecli_40_60, flag_agecli_sup_60  )


#AGE_VEH
segments_age_ve<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == 0){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ageve_neuf =segments_age_ve(age_ve)

segments_age_ve<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 0 & x[i] <= 25 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ageve_0_25 =segments_age_ve(age_ve)

segments_age_ve<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 25 & x[i] <= 50 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ageve_25_50 =segments_age_ve(age_ve)

segments_age_ve<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 50 ){ x[i] = 1 }
```

```
    else { x[i] = 0 }
  }
  return (x)
}
flag_ageve_sup_50 =segments_age_ve(age_ve)


flag_ageve = cbind( flag_ageve_neuf, flag_ageve_0_25, flag_ageve_25_50, flag_ageve_sup_50 )


# PRIX_VEH
segments_prix_ve<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] <= 10000 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prix_ve_inf_10k =segments_prix_ve(prix_ve)

segments_prix_ve<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 10000 & x[i] <= 15000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prix_ve_10k_15k =segments_prix_ve(prix_ve)

segments_prix_ve<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 15000 & x[i] <= 20000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prix_ve_15k_20k =segments_prix_ve(prix_ve)

segments_prix_ve<-function(x){
  for (i in 1:nrow(data)){
    if ( x[i] > 20000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prix_ve_sup_20k =segments_prix_ve(prix_ve)


flag_prix_ve = cbind( flag_prix_ve_inf_10k, flag_prix_ve_10k_15k, flag_prix_ve_15k_20k, flag_prix_ve_su



# PC_MENS
```

```
pc_mens=(mens/prix_ve)*100
segments_pc_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 1.2581){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_mens_inf_1.2581 = segments_pc_mens(pc_mens)

segments_pc_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 1.2581 & x[i] <= 1.5149){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_mens_1.2581_1.5149 = segments_pc_mens(pc_mens) #1.51=1er quantile

segments_pc_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 1.5149 & x[i] <= 1.7745 ){ x[i] = 1 } # moyenne 1.7745
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_mens_1.5149_1.7745= segments_pc_mens(pc_mens)

segments_pc_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 1.7745 ){ x[i] = 1 } # moyenne 1.7745
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_mens_sup_1.7745= segments_pc_mens(pc_mens)


flag_pc_mens = cbind( flag_pc_mens_inf_1.2581,flag_pc_mens_1.2581_1.5149,flag_pc_mens_1.5149_1.7745, fla




#ASSUR
segments_noassur<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == 0  ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_noassur = segments_noassur(assur)
```

```
segments_assur<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] >0 & x[i] <= 500 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_assur_inf_500 = segments_assur(assur)

segments_assur<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 500 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_assur_sup_500 = segments_assur(assur)

flag_assur = cbind( flag_noassur, flag_assur_inf_500, flag_assur_sup_500)



# VR_BALL

segments_ball<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==0 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_noball = segments_ball(vr_ball)

segments_ball<-function(x){       #summary(vr_ball)  moy= 9390
  for (i in 1:nrow(data)){
    if (x[i] >0  & x[i] <= 10000 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ball_inf_10000 = segments_ball(vr_ball)

segments_ball<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 10000 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_ball_sup_10000 = segments_ball(vr_ball)


flag_ball = cbind( flag_noball, flag_ball_inf_10000, flag_ball_sup_10000 )
```

```r
# MT_APPORT

segments_apport<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 5000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_apport_inf_5000 = segments_apport(apport)


segments_apport<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 5000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_apport_sup_5000 = segments_apport(apport)


flag_apport = cbind( flag_apport_inf_5000, flag_apport_sup_5000 )




# MARQUE
flag_marque_RENAULT = rep(0, nrow(data))
flag_marque_DACIA = rep(0, nrow(data))
flag_marque_NISSAN = rep(0, nrow(data))
flag_marque_AUTRE = rep(0, nrow(data))
flag_marque = cbind( flag_marque_RENAULT,flag_marque_DACIA, flag_marque_NISSAN, flag_marque_AUTRE )


#DUREE_CONTRAT
segments_duree<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 24){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_duree_inf_24 = segments_duree(duree)

segments_duree<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 24 & x[i] <= 36){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
```

```
}
flag_duree_24_36 = segments_duree(duree)

segments_duree<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 36){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_duree_sup_36 = segments_duree(duree)


flag_duree = cbind( flag_duree_inf_24, flag_duree_24_36, flag_duree_sup_36 )



#PREST

segments_noprest<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==0){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_noprest = segments_noprest(prest)

segments_prest<-function(x){
  for (i in 1:nrow(data)){
    if (x[i]>0 & x[i] <= 140 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}

flag_prest_inf_140 = segments_prest(prest)

segments_prest<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 140 & x[i] <= 400 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_prest_140_400 = segments_prest(prest)

segments_prest<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 400 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
```

```
}
flag_prest_sup_400 = segments_prest(prest)


flag_prest = cbind( flag_prest_inf_140, flag_prest_140_400, flag_prest_sup_400, flag_noprest  )



#MODE_LOGT
segments_loge<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==  1 | x[i]==3 | x[i]==4){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_loge_nonprop = segments_loge(loge)

segments_loge<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==  2 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_loge_proprietaire = segments_loge(loge)

flag_loge = cbind(flag_loge_proprietaire, flag_loge_nonprop)



#PC_APPO
segments_pc_appo<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 13){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_appo_inf_13 = segments_pc_appo(pc_appo)

segments_pc_appo<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 13 & x[i] <= 27){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_appo_13_27 = segments_pc_appo(pc_appo)

segments_pc_appo<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 27 & x[i] <= 36){ x[i] = 1 }
```

```
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_appo_27_36 = segments_pc_appo(pc_appo)

segments_pc_appo<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 36){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_pc_appo_sup_36 = segments_pc_appo(pc_appo)


flag_pc_appo = cbind( flag_pc_appo_inf_13, flag_pc_appo_13_27, flag_pc_appo_27_36, flag_pc_appo_sup_36 )


#SIT_FAM
segments_sit_fam<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == 1 | x[i]==3){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_sit_fam_marie_div = segments_sit_fam(sit_fam)

segments_sit_fam<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == 2 | x[i] == 4 | x[i] == 5 | x[i] == 11){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_sit_fam_autre = segments_sit_fam(sit_fam)

flag_sit_fam = cbind( flag_sit_fam_marie_div, flag_sit_fam_autre)


#MENS
segments_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 200){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_mens_inf_200 = segments_mens(mens)

segments_mens<-function(x){
  for (i in 1:nrow(data)){
```

```
    if (x[i] > 200 & x[i] <= 300){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_mens_200_300 = segments_mens(mens)

segments_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 300 & x[i] <= 400){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_mens_300_400 = segments_mens(mens)

segments_mens<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 400){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_mens_sup_400 = segments_mens(mens)


flag_mens = cbind( flag_mens_inf_200, flag_mens_200_300, flag_mens_300_400, flag_mens_sup_400 )

#BENEF

delta<-function(x){
  for (i in 1:nrow(data)){
  x[i]= (duree[i]*mens[i])-finance[i]+vr_ball[i]}
return (x)
}
benef=delta(vr_ball)

segments_benef<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 1000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_benef_inf_1k = segments_benef(benef)

segments_benef<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 1000 & x[i] <= 2000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
```

```
flag_benef_1k_2k = segments_benef(benef)

segments_benef<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 2000 & x[i] <= 3000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_benef_2k_3k = segments_benef(benef)

segments_benef<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 3000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_benef_sup_3k = segments_benef(benef)


flag_benef = cbind( flag_benef_inf_1k, flag_benef_1k_2k, flag_benef_2k_3k, flag_benef_sup_3k)


# FINANCE
segments_finance<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 10000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_finance_inf_10k = segments_finance(finance)

segments_finance<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 10000 & x[i] <= 13000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_finance_10k_13k = segments_finance(finance)

segments_finance<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 13000 & x[i] <= 16000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_finance_13k_16k = segments_finance(finance)

segments_finance<-function(x){
```

```
  for (i in 1:nrow(data)){
    if (x[i] > 16000){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_finance_sup_16k = segments_finance(finance)


flag_finance = cbind( flag_finance_inf_10k, flag_finance_10k_13k, flag_finance_13k_16k, flag_finance_su

# anciennete-rci
segments_anc<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] ==-1){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_NA_anc = segments_anc(ancien_rci)

segments_anc<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] <= 2){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_anc_inf_2 = segments_anc(ancien_rci)

segments_anc<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] > 2 ){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_anc_sup_2 = segments_anc(ancien_rci)

flag_anc = cbind( flag_NA_anc, flag_anc_inf_2, flag_anc_sup_2 )




#NO
segments_NO<-function(x){
  for (i in 1:nrow(data)){
    if (x[i] == 0){ x[i] = 1 }
    else { x[i] = 0 }
  }
  return (x)
}
flag_NO = segments_NO(age_ve)
```

```
flag_NO <- as.numeric(flag_NO)



features=cbind(duree, anc_emploi, sit_fam, loge, age_ve,prix_ve, apport, finance,  mens, vr_ball, prest


flag=cbind(flag_emploi, flag_agecli, flag_ageve, flag_prix_ve, flag_pc_mens, flag_assur, flag_ball, flag

def <- def-1

train_total <-as.data.frame(cbind(def,features,flag))


write.csv(train_total,"train_reech.csv", row.names = FALSE)
```

LOGISTIQUE REECHANTILLONNEE

```
remove(list=objects())
train <- read.csv2('train_reech.csv', sep=",",header=TRUE)
test <- read.csv2('test.csv', sep=",",header=TRUE)



train[,1]=as.numeric(train[,1])
train[,2]=as.numeric(train[,2])
train[,3]=as.numeric(train[,3])
train[,4]=as.numeric(train[,4])
train[,5]=as.numeric(train[,5])
train[,6]=as.numeric(train[,6])
train[,7]=as.numeric(train[,7])
train[,8]=as.numeric(train[,8])
train[,9]=as.numeric(train[,9])
train[,10]=as.numeric(train[,10])
train[,11]=as.numeric(train[,11])
train[,12]=as.numeric(train[,12])
train[,13]=as.numeric(train[,13])
train[,14]=as.numeric(train[,14])
train[,15]=as.numeric(train[,15])
train[,16]=as.numeric(train[,16])
train[,17]=as.numeric(train[,17])
train[,18]=as.numeric(train[,18])

test[,1]=as.numeric(test[,1])
test[,2]=as.numeric(test[,2])
test[,3]=as.numeric(test[,3])
test[,4]=as.numeric(test[,4])
test[,5]=as.numeric(test[,5])
test[,6]=as.numeric(test[,6])
test[,7]=as.numeric(test[,7])
test[,8]=as.numeric(test[,8])
test[,9]=as.numeric(test[,9])
test[,10]=as.numeric(test[,10])
```

```
test[,11]=as.numeric(test[,11])
test[,12]=as.numeric(test[,12])
test[,13]=as.numeric(test[,13])
test[,14]=as.numeric(test[,14])
test[,15]=as.numeric(test[,15])
test[,16]=as.numeric(test[,16])
test[,17]=as.numeric(test[,17])
test[,18]=as.numeric(test[,18])
```

LOGISTIQUE

```
modele <- glm(def~ (prix_ve + anc_emploi + flag_agecli_40_60 + age_ve + prest + flag_sit_fam_marie_div +



coef <- as.numeric(coef(modele))
score <- predict.glm(modele, newdata=test[,-1])
```

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

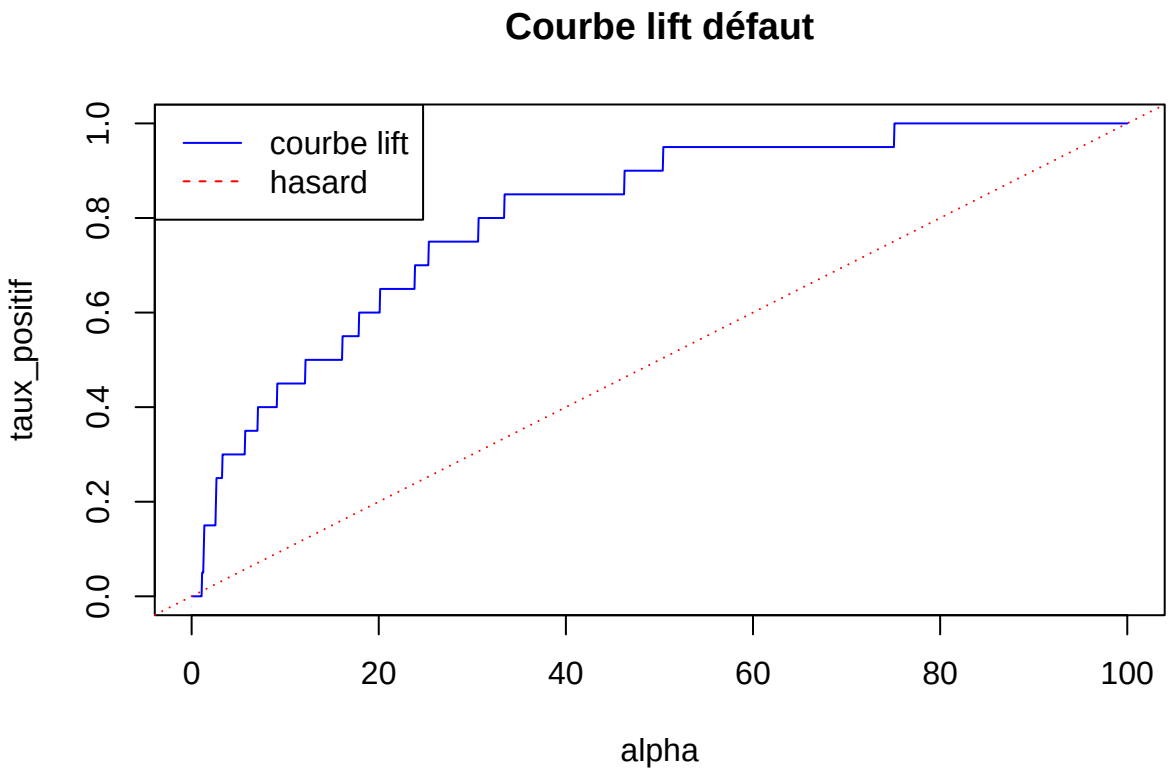
```
proba <- invlogit(score)

id <- as.numeric(as.character(rownames(test)))
data_predict <- as.data.frame(cbind(proba,test[1]))

scoring <- as.data.frame(cbind(proba, test[1]))
write.csv(scoring, "scoring_log_reech.csv")
```

PERFORMANCES

Courbe lift défaut On représente ici l'évolution du lift par rapport au score de défaut en fonction de l'alpha(% des données) choisi.

```
data_perf <- data_predict[order(-proba),]

taux_positif <- rep(0,nrow(data_perf))
if (data_perf[1,2]==1){
  taux_positif <- 1
}

for (i in 2:nrow(data_perf)){
  if (data_perf[i,2] ==1){
      taux_positif[i] <- taux_positif[i-1] +1
  }
  else{
    taux_positif[i] <- taux_positif[i-1]
  }
}
taux_positif <- taux_positif/ sum(data_perf[,2])

alpha <- rep(0,nrow(data_perf))
for (i in 1:nrow(data_perf)){
  alpha[i]<- i / nrow(data_perf) *100
}

plot(taux_positif~alpha, type='l', col="blue", main = "Courbe lift défaut")
```

```
abline(a=0,b=0.01, col='red', lty ='dotted')
legend('topleft', legend=c("courbe lift", "hasard"), col=c("blue","red"), lty=1:2)
```



Courbe lift octroi On représente ici l'évolution du lift par rapport au score d'octroi en fonction de l'alpha(% des données) choisi. On observe un lift equivalent au hasard, ce qui semble logique car avec un taux cible >99% (le non défaut), trier au hasard donne déja un bon score.

```
data_perf_2 <- data_predict[order(proba),]

taux_negatif <- rep(0,nrow(data_perf_2))
if (data_perf_2[1,2]==0){
  taux_negatif <- 1
}

for (i in 2:nrow(data_perf)){
  if (data_perf_2[i,2] ==0){
      taux_negatif[i] <- taux_negatif[i-1] +1
  }
  else{
    taux_negatif[i] <- taux_negatif[i-1]
  }
}
taux_negatif <- taux_negatif/ (nrow(data_perf) -sum(data_perf_2[,2]))

alpha <- rep(0,nrow(data_perf_2))
for (i in 1:nrow(data_perf_2)){
  alpha[i]<- i / nrow(data_perf_2) *100
}
```

```
plot(taux_negatif~alpha, type='l', col="blue", main = "Courbe lift octroi")
abline(a=0,b=0.01, col='red', lty ='dotted')
legend('topleft', legend=c("courbe lift", "hasard"), col=c("blue","red"), lty=1:2)
```

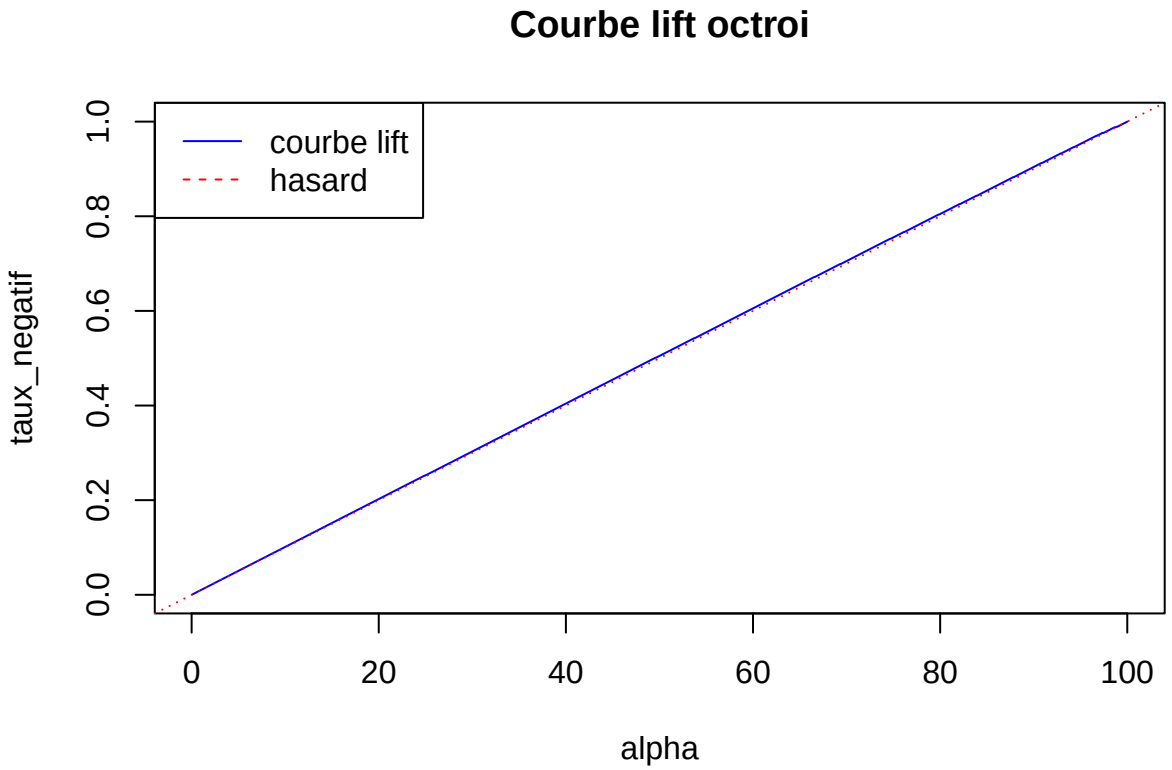


Tableau lift defaut

```
def=test$def
tab1=order(proba,decreasing=TRUE)

i=1
s=0
j=1
k=1
nombre_positif=rep(0,10)
effectif=rep(0,10)
while (i<=1521){
  if (def[tab1[i]]==1){s=s+1}
  if (i%%169==0){
    nombre_positif[j]=s
    effectif[j]=k
    s=0
    j=j+1
    k=0

  }
  i=i+1
  k=k+1
}
effectif[10]=1692-1521
s=0
```

```
for (i in 1521:1692){
  if (def[tab1[i]]==1){s=s+1}
}
nombre_positif[10]=s

taux_cible=nombre_positif/effectif
taux_cible_total=sum(def)/length(def)

alpha=seq(10,100,10)
alpha_lift=taux_cible/taux_cible_total

alpha_lift_cumul=rep(0,10)
alpha_lift_cumul[1]=alpha_lift[1]
mem1=nombre_positif[1]
mem2=effectif[1]
for (i in 2:10){
 alpha_lift_cumul[i]=((mem1+nombre_positif[i])/(mem2+effectif[i]))*(1/taux_cible_total)
 mem1=mem1+nombre_positif[i]
 mem2=mem2+effectif[i]
}

tab5=cbind(alpha,effectif,nombre_positif,taux_cible,alpha_lift,alpha_lift_cumul)
tableau_lift=round(tab5,2)
tableau_lift
```

```
##       alpha effectif nombre_positif taux_cible alpha_lift alpha_lift_cumul
##  [1,]    10      169              9       0.05       4.51             4.51
##  [2,]    20      169              3       0.02       1.50             3.00
##  [3,]    30      169              3       0.02       1.50             2.50
##  [4,]    40      169              2       0.01       1.00             2.13
##  [5,]    50      169              1       0.01       0.50             1.80
##  [6,]    60      169              1       0.01       0.50             1.59
##  [7,]    70      169              0       0.00       0.00             1.36
##  [8,]    80      169              1       0.01       0.50             1.25
##  [9,]    90      169              0       0.00       0.00             1.11
## [10,]   100      171              0       0.00       0.00             1.00
```

Tableau lift octroi

```
def=test$def
tab1=order(proba,decreasing=F)

i=1
s=0
j=1
k=1
nombre_positif=rep(0,10)
effectif=rep(0,10)
while (i<=1521){
  if (def[tab1[i]]==0){s=s+1}
  if (i%%169==0){
    nombre_positif[j]=s
    effectif[j]=k
    s=0
    j=j+1
```

```
    k=0

  }
  i=i+1
  k=k+1
}
effectif[10]=1692-1521
s=0
for (i in 1521:1692){
  if (def[tab1[i]]==0){s=s+1}
}
nombre_positif[10]=s

taux_cible=nombre_positif/effectif
taux_cible_total=(1692-sum(def))/length(def)

alpha=seq(10,100,10)
alpha_lift=taux_cible/taux_cible_total

alpha_lift_cumul=rep(0,10)
alpha_lift_cumul[1]=alpha_lift[1]
mem1=nombre_positif[1]
mem2=effectif[1]
for (i in 2:10){
 alpha_lift_cumul[i]=((mem1+nombre_positif[i])/(mem2+effectif[i]))*(1/taux_cible_total)
 mem1=mem1+nombre_positif[i]
 mem2=mem2+effectif[i]
}

tab5=cbind(alpha,effectif,nombre_positif,taux_cible,alpha_lift,alpha_lift_cumul)
tableau_lift=round(tab5,2)
tableau_lift
```

```
##       alpha effectif nombre_positif taux_cible alpha_lift alpha_lift_cumul
##  [1,]    10      169            169       1.00       1.01             1.01
##  [2,]    20      169            169       1.00       1.01             1.01
##  [3,]    30      169            168       0.99       1.01             1.01
##  [4,]    40      169            169       1.00       1.01             1.01
##  [5,]    50      169            168       0.99       1.01             1.01
##  [6,]    60      169            168       0.99       1.01             1.01
##  [7,]    70      169            167       0.99       1.00             1.01
##  [8,]    80      169            166       0.98       0.99             1.01
##  [9,]    90      169            166       0.98       0.99             1.00
## [10,]   100      171            163       0.95       0.96             1.00
```

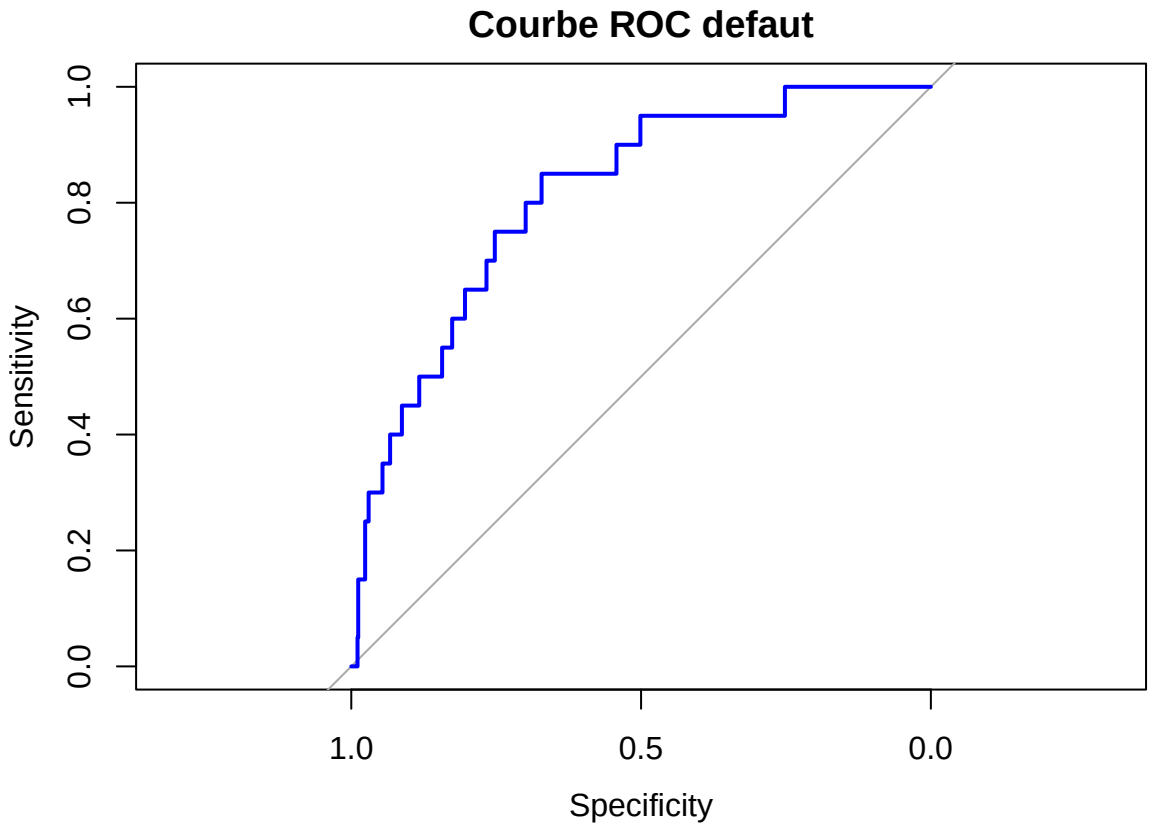
Courbes ROC et AUC On observe les courbes ROC de défaut et d'octroi, ainsi que l'AUC (aire sous la courbe ROC). AUC de 0.5 = comme le hasard AUC de 1 = score parfait

```
ROC_defaut <- roc(data_perf, response = def, predictor = proba)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(ROC_defaut, main ="Courbe ROC defaut", col="blue")
```

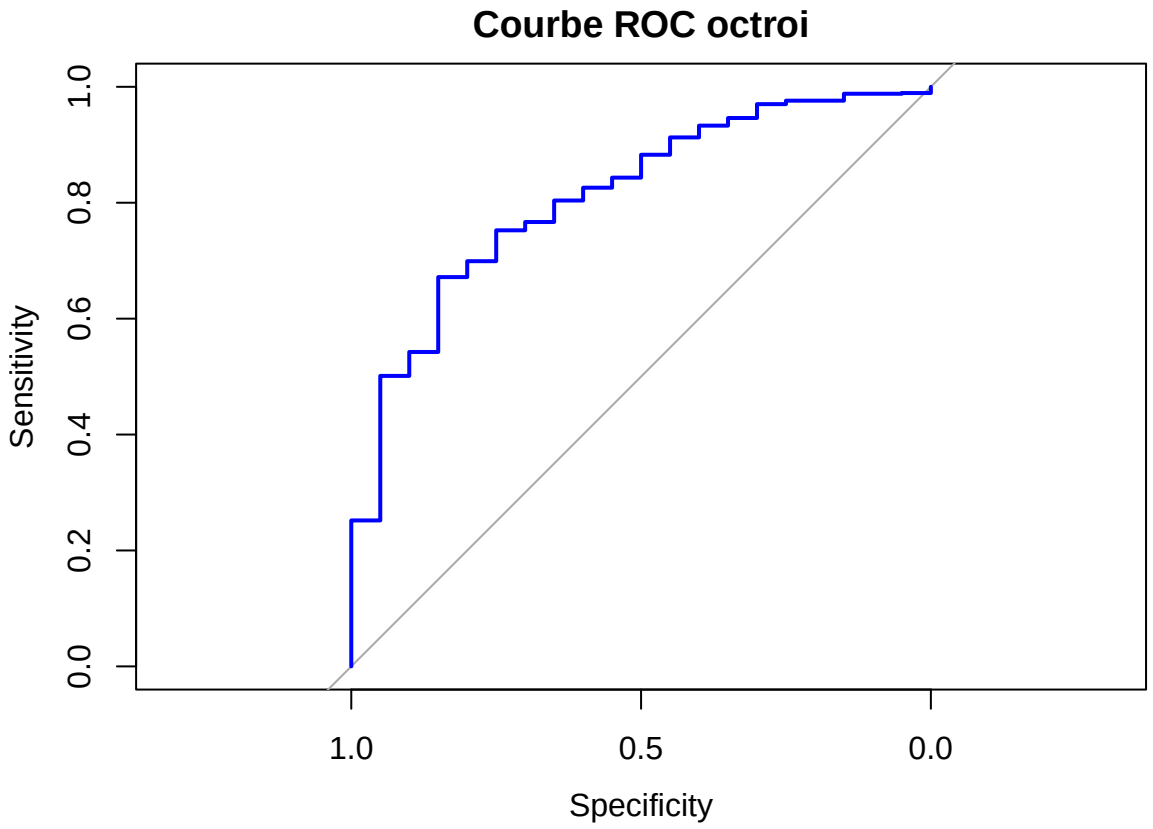



```
data_perf_3 <- data_perf_2
for (i in 1:nrow(data_perf_3)){
  if (data_perf_3[i,2]==1){
    data_perf_3[i,2]<-0
  }
  else{
    data_perf_3[i,2]<-1
  }
}
ROC_octroi <- roc(data_perf_3, response = def, predictor = proba)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
plot(ROC_octroi, main="Courbe ROC octroi", col="blue")
```




```
AUC<- auc(ROC_defaut)
print(AUC)
```

```
## Area under the curve: 0.811
```

```
#proba de placer un 1 avant un 0

GINI <- 2*AUC-1
print(GINI)
```

```
## [1] 0.6220694
```

```
# % de mieux par rapport au hasard
```

LOGISTIQUE NON REECHANTILLONNEE

```
remove(list=objects())
train <- read.csv2('train_non_reech.csv', sep=",",header=TRUE)
test <- read.csv2('test.csv', sep=",",header=TRUE)



train[,1]=as.numeric(train[,1])
train[,2]=as.numeric(train[,2])
train[,3]=as.numeric(train[,3])
train[,4]=as.numeric(train[,4])
train[,5]=as.numeric(train[,5])
train[,6]=as.numeric(train[,6])
```

```
train[,7]=as.numeric(train[,7])
train[,8]=as.numeric(train[,8])
train[,9]=as.numeric(train[,9])
train[,10]=as.numeric(train[,10])
train[,11]=as.numeric(train[,11])
train[,12]=as.numeric(train[,12])
train[,13]=as.numeric(train[,13])
train[,14]=as.numeric(train[,14])
train[,15]=as.numeric(train[,15])
train[,16]=as.numeric(train[,16])
train[,17]=as.numeric(train[,17])
train[,18]=as.numeric(train[,18])

test[,1]=as.numeric(test[,1])
test[,2]=as.numeric(test[,2])
test[,3]=as.numeric(test[,3])
test[,4]=as.numeric(test[,4])
test[,5]=as.numeric(test[,5])
test[,6]=as.numeric(test[,6])
test[,7]=as.numeric(test[,7])
test[,8]=as.numeric(test[,8])
test[,9]=as.numeric(test[,9])
test[,10]=as.numeric(test[,10])
test[,11]=as.numeric(test[,11])
test[,12]=as.numeric(test[,12])
test[,13]=as.numeric(test[,13])
test[,14]=as.numeric(test[,14])
test[,15]=as.numeric(test[,15])
test[,16]=as.numeric(test[,16])
test[,17]=as.numeric(test[,17])
test[,18]=as.numeric(test[,18])
```

LOGISTIQUE

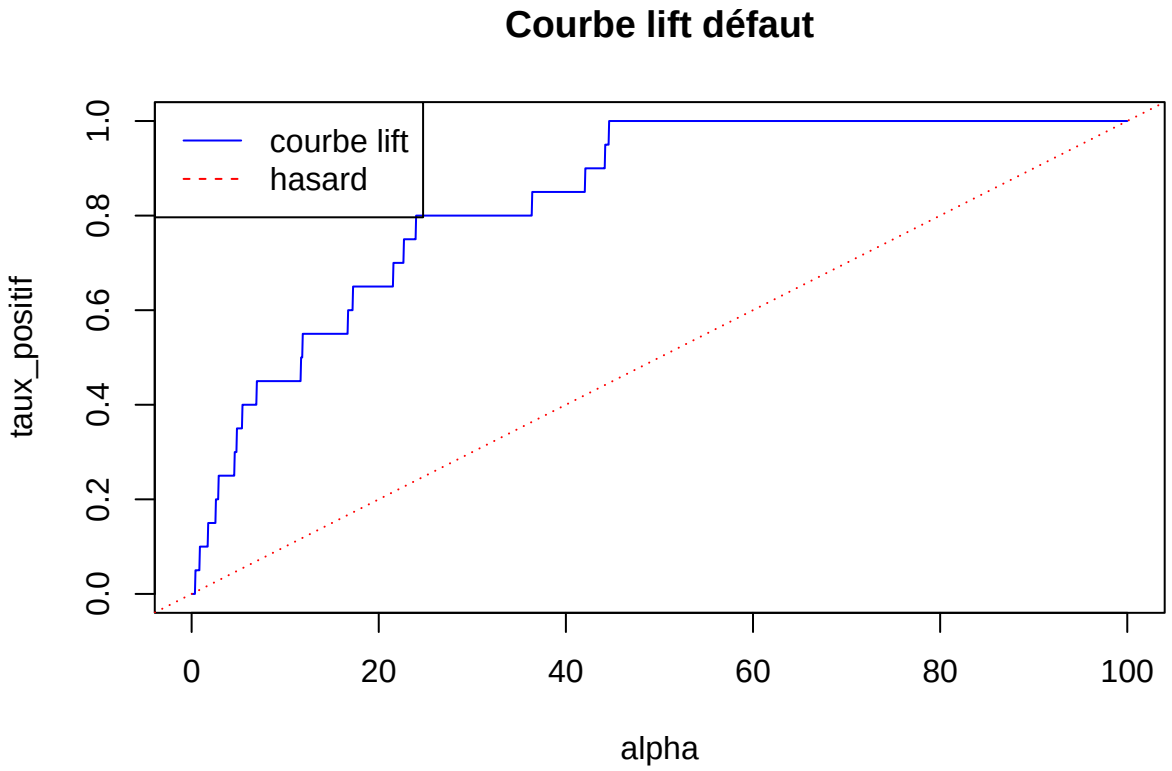
```
modele <- glm(def~ (prix_ve + anc_emploi + flag_agecli_40_60 + age_ve + prest + flag_sit_fam_marie_div +



coef <- as.numeric(coef(modele))
score <- predict.glm(modele, newdata=test[,-1])
proba <- invlogit(score)

id <- as.numeric(as.character(rownames(test)))
data_predict <- as.data.frame(cbind(proba,test[1]))

scoring <- as.data.frame(cbind(proba, test[1]))
write.csv(scoring, "scoring_log_non_reech.csv")
```

PERFORMANCES

Courbe lift défaut On représente ici l'évolution du lift par rapport au score de défaut en fonction de l'alpha(% des données) choisi.

```
data_perf <- data_predict[order(-proba),]

taux_positif <- rep(0,nrow(data_perf))
```

```r
if (data_perf[1,2]==1){
  taux_positif <- 1
}

for (i in 2:nrow(data_perf)){
  if (data_perf[i,2] ==1){
      taux_positif[i] <- taux_positif[i-1] +1
  }
  else{
    taux_positif[i] <- taux_positif[i-1]
  }
}
taux_positif <- taux_positif/ sum(data_perf[,2])

alpha <- rep(0,nrow(data_perf))
for (i in 1:nrow(data_perf)){
  alpha[i]<- i / nrow(data_perf) *100
}

plot(taux_positif~alpha, type='l', col="blue", main = "Courbe lift défaut")
abline(a=0,b=0.01, col='red', lty ='dotted')
legend('topleft', legend=c("courbe lift", "hasard"), col=c("blue","red"), lty=1:2)
```



Courbe lift octroi On représente ici l'évolution du lift par rapport au score d'octroi en fonction de l'alpha(% des données) choisi. On observe un lift equivalent au hasard, ce qui semble logique car avec un taux cible >99% (le non défaut), trier au hasard donne déja un bon score.

```r
data_perf_2 <- data_predict[order(proba),]

taux_negatif <- rep(0,nrow(data_perf_2))
```

```r
if (data_perf_2[1,2]==0){
  taux_negatif <- 1
}

for (i in 2:nrow(data_perf)){
  if (data_perf_2[i,2] ==0){
      taux_negatif[i] <- taux_negatif[i-1] +1
  }
  else{
    taux_negatif[i] <- taux_negatif[i-1]
  }
}
taux_negatif <- taux_negatif/ (nrow(data_perf) -sum(data_perf_2[,2]))

alpha <- rep(0,nrow(data_perf_2))
for (i in 1:nrow(data_perf_2)){
  alpha[i]<- i / nrow(data_perf_2) *100
}


plot(taux_negatif~alpha, type='l', col="blue", main = "Courbe lift octroi")
abline(a=0,b=0.01, col='red', lty ='dotted')
legend('topleft', legend=c("courbe lift", "hasard"), col=c("blue","red"), lty=1:2)
```

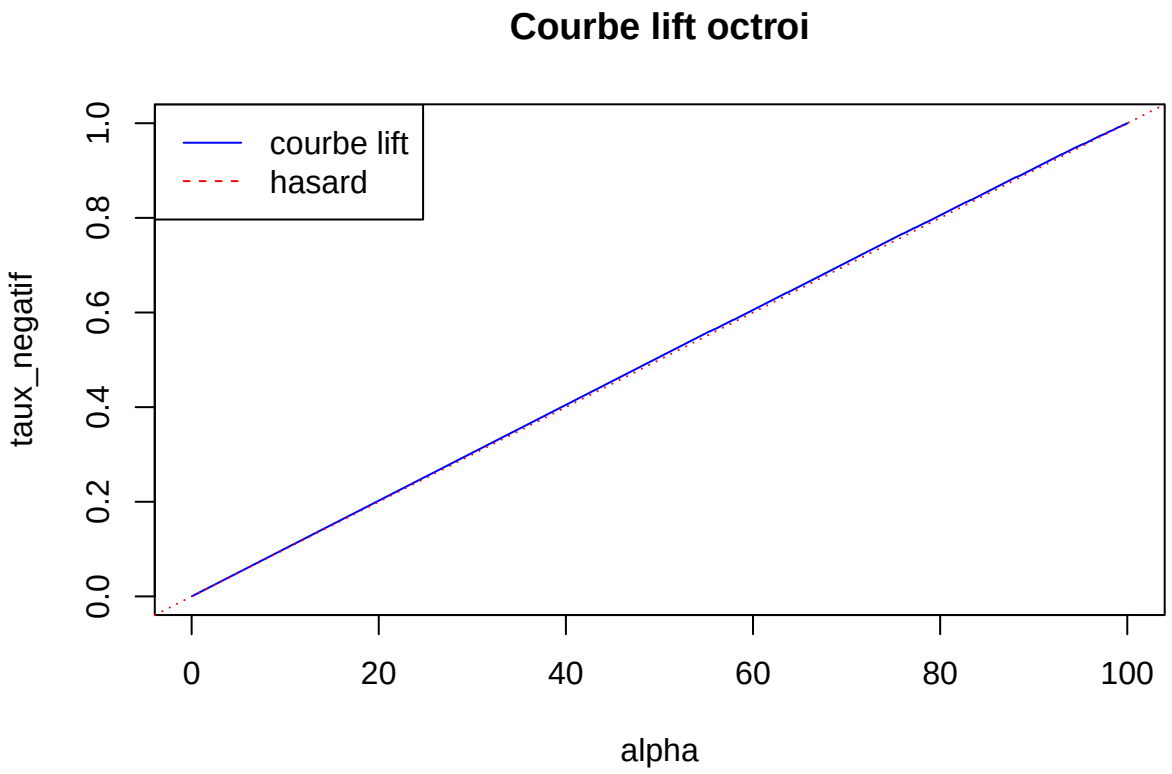


Tableau lift defaut

```r
def=test$def
tab1=order(proba,decreasing=TRUE)

i=1
```

```
s=0
j=1
k=1
nombre_positif=rep(0,10)
effectif=rep(0,10)
while (i<=1521){
  if (def[tab1[i]]==1){s=s+1}
  if (i%%169==0){
    nombre_positif[j]=s
    effectif[j]=k
    s=0
    j=j+1
    k=0

  }
  i=i+1
  k=k+1
}
effectif[10]=1692-1521
s=0
for (i in 1521:1692){
  if (def[tab1[i]]==1){s=s+1}
}
nombre_positif[10]=s

taux_cible=nombre_positif/effectif
taux_cible_total=sum(def)/length(def)

alpha=seq(10,100,10)
alpha_lift=taux_cible/taux_cible_total

alpha_lift_cumul=rep(0,10)
alpha_lift_cumul[1]=alpha_lift[1]
mem1=nombre_positif[1]
mem2=effectif[1]
for (i in 2:10){
 alpha_lift_cumul[i]=((mem1+nombre_positif[i])/(mem2+effectif[i]))*(1/taux_cible_total)
 mem1=mem1+nombre_positif[i]
 mem2=mem2+effectif[i]
}

tab5=cbind(alpha,effectif,nombre_positif,taux_cible,alpha_lift,alpha_lift_cumul)
tableau_lift=round(tab5,2)
tableau_lift
```

```
##       alpha effectif nombre_positif taux_cible alpha_lift alpha_lift_cumul
##  [1,]    10      169              9       0.05       4.51             4.51
##  [2,]    20      169              4       0.02       2.00             3.25
##  [3,]    30      169              3       0.02       1.50             2.67
##  [4,]    40      169              1       0.01       0.50             2.13
##  [5,]    50      169              3       0.02       1.50             2.00
##  [6,]    60      169              0       0.00       0.00             1.67
##  [7,]    70      169              0       0.00       0.00             1.43
##  [8,]    80      169              0       0.00       0.00             1.25
```

```
##  [9,]    90      169              0       0.00       0.00             1.11
## [10,]   100      171              0       0.00       0.00             1.00
```

Tableau lift octroi

```
def=test$def
tab1=order(proba,decreasing=F)

i=1
s=0
j=1
k=1
nombre_positif=rep(0,10)
effectif=rep(0,10)
while (i<=1521){
  if (def[tab1[i]]==0){s=s+1}
  if (i%%169==0){
    nombre_positif[j]=s
    effectif[j]=k
    s=0
    j=j+1
    k=0

  }
  i=i+1
  k=k+1
}
effectif[10]=1692-1521
s=0
for (i in 1521:1692){
  if (def[tab1[i]]==0){s=s+1}
}
nombre_positif[10]=s

taux_cible=nombre_positif/effectif
taux_cible_total=(1692-sum(def))/length(def)

alpha=seq(10,100,10)
alpha_lift=taux_cible/taux_cible_total

alpha_lift_cumul=rep(0,10)
alpha_lift_cumul[1]=alpha_lift[1]
mem1=nombre_positif[1]
mem2=effectif[1]
for (i in 2:10){
 alpha_lift_cumul[i]=((mem1+nombre_positif[i])/(mem2+effectif[i]))*(1/taux_cible_total)
 mem1=mem1+nombre_positif[i]
 mem2=mem2+effectif[i]
}

tab5=cbind(alpha,effectif,nombre_positif,taux_cible,alpha_lift,alpha_lift_cumul)
tableau_lift=round(tab5,2)
tableau_lift
```

```
##       alpha effectif nombre_positif taux_cible alpha_lift alpha_lift_cumul
```

```
##  [1,]    10      169           169       1.00       1.01           1.01
##  [2,]    20      169           169       1.00       1.01           1.01
##  [3,]    30      169           169       1.00       1.01           1.01
##  [4,]    40      169           169       1.00       1.01           1.01
##  [5,]    50      169           169       1.00       1.01           1.01
##  [6,]    60      169           166       0.98       0.99           1.01
##  [7,]    70      169           168       0.99       1.01           1.01
##  [8,]    80      169           166       0.98       0.99           1.01
##  [9,]    90      169           165       0.98       0.99           1.00
## [10,]   100      171           163       0.95       0.96           1.00
```

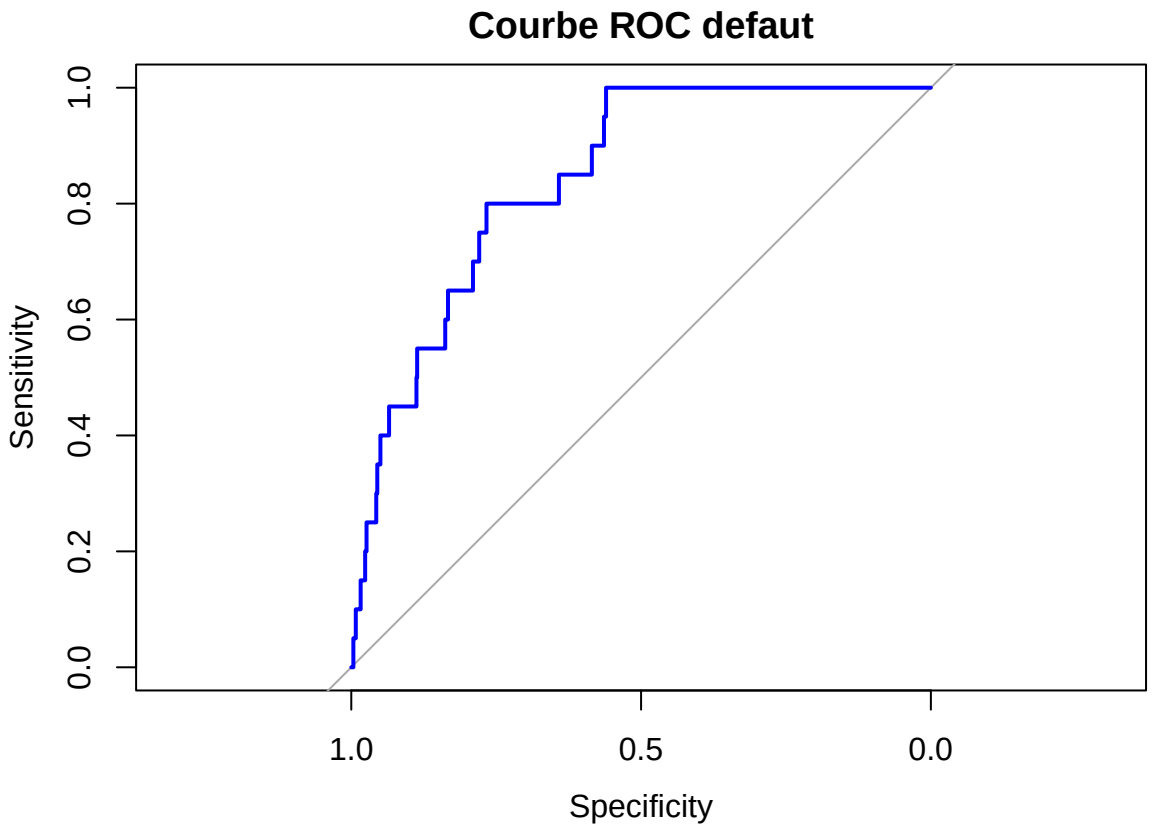
Courbes ROC et AUC On observe les courbes ROC de défaut et d'octroi, ainsi que l'AUC (aire sous la courbe ROC). AUC de 0.5 = comme le hasard AUC de 1 = score parfait

```
ROC_defaut <- roc(data_perf, response = def, predictor = proba)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(ROC_defaut, main ="Courbe ROC defaut", col="blue")
```



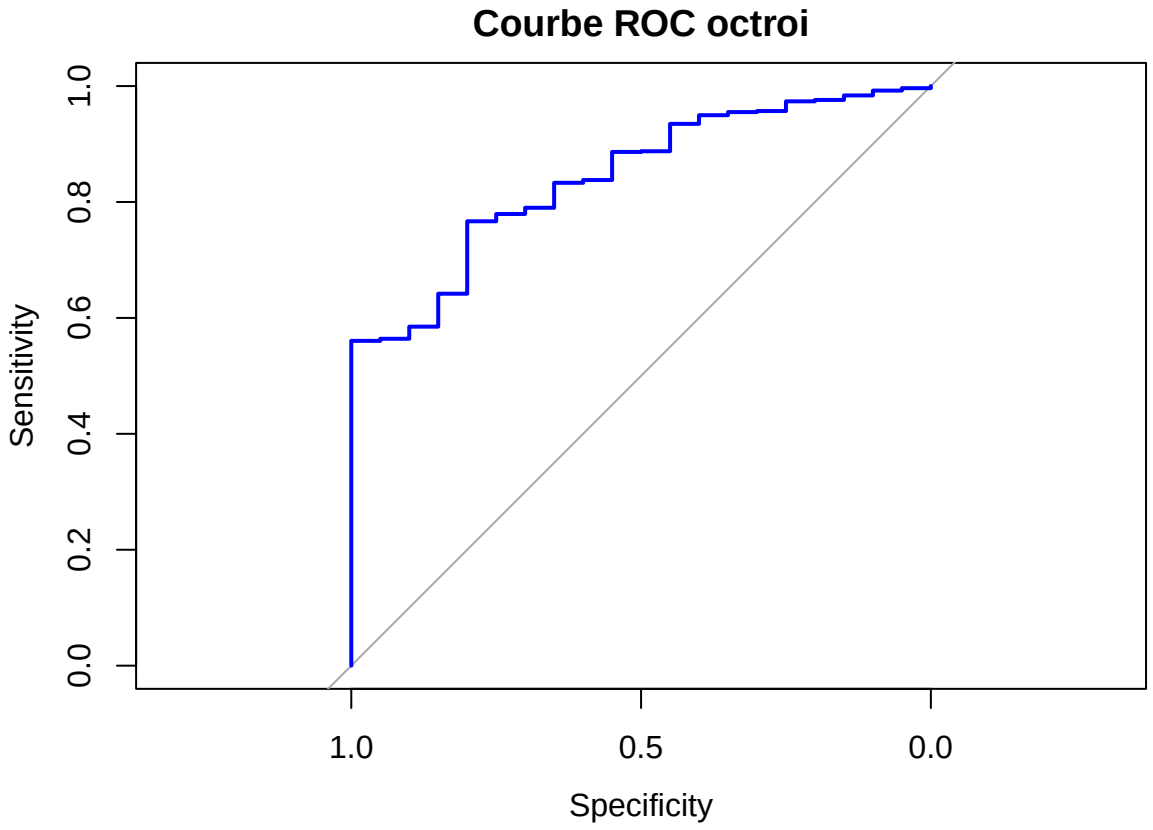

```
data_perf_3 <- data_perf_2
for (i in 1:nrow(data_perf_3)){
  if (data_perf_3[i,2]==1){
    data_perf_3[i,2]<-0
  }
  else{
    data_perf_3[i,2]<-1
  }
```

```
}
ROC_octroi <- roc(data_perf_3, response = def, predictor = proba)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
plot(ROC_octroi, main="Courbe ROC octroi", col="blue")
```




```
AUC<- auc(ROC_defaut)
print(AUC)
```

```
## Area under the curve: 0.8426
```

```
#proba de placer un 1 avant un 0

GINI <- 2*AUC-1
print(GINI)
```

```
## [1] 0.6851077
```

```
# % de mieux par rapport au hasard
```

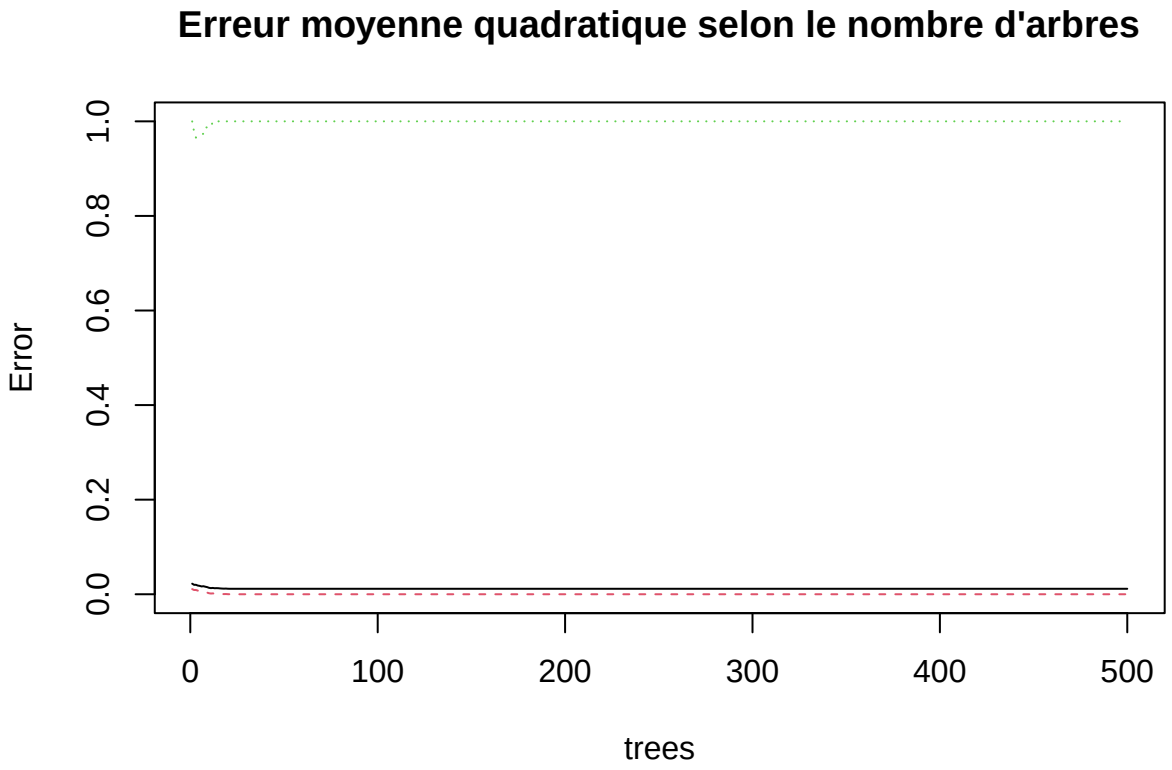
RANDOM FOREST

```
remove(list=objects())
train <- read.csv2('train_non_reech.csv', sep=",",header=TRUE)
test <- read.csv2('test.csv', sep=",",header=TRUE)
```

```
train[,1]=as.numeric(train[,1])
train[,2]=as.numeric(train[,2])
train[,3]=as.numeric(train[,3])
train[,4]=as.numeric(train[,4])
train[,5]=as.numeric(train[,5])
train[,6]=as.numeric(train[,6])
train[,7]=as.numeric(train[,7])
train[,8]=as.numeric(train[,8])
train[,9]=as.numeric(train[,9])
train[,10]=as.numeric(train[,10])
train[,11]=as.numeric(train[,11])
train[,12]=as.numeric(train[,12])
train[,13]=as.numeric(train[,13])
train[,14]=as.numeric(train[,14])
train[,15]=as.numeric(train[,15])
train[,16]=as.numeric(train[,16])
train[,17]=as.numeric(train[,17])
train[,18]=as.numeric(train[,18])

test[,1]=as.numeric(test[,1])
test[,2]=as.numeric(test[,2])
test[,3]=as.numeric(test[,3])
test[,4]=as.numeric(test[,4])
test[,5]=as.numeric(test[,5])
test[,6]=as.numeric(test[,6])
test[,7]=as.numeric(test[,7])
test[,8]=as.numeric(test[,8])
test[,9]=as.numeric(test[,9])
test[,10]=as.numeric(test[,10])
test[,11]=as.numeric(test[,11])
test[,12]=as.numeric(test[,12])
test[,13]=as.numeric(test[,13])
test[,14]=as.numeric(test[,14])
test[,15]=as.numeric(test[,15])
test[,16]=as.numeric(test[,16])
test[,17]=as.numeric(test[,17])
test[,18]=as.numeric(test[,18])
```

RANDOM FOREST

```
set.seed(1234)
train$def <- as.factor(train$def)
modele <- randomForest(def ~., data=train, mtry=15, nodesize=5, ntree=500, type="regression")

prediction <- predict(modele, test, type="prob")
proba <- prediction[,2]
id <- as.numeric(as.character(rownames(test)))
data_predict <- as.data.frame(cbind(proba, test[1]))
write.csv(data_predict, "scoring_rf_non_reech.csv")
plot(modele, type="l", main="Erreur moyenne quadratique selon le nombre d'arbres")
```



OPTIMISATION DU MODELE ET DES PARAMETRES

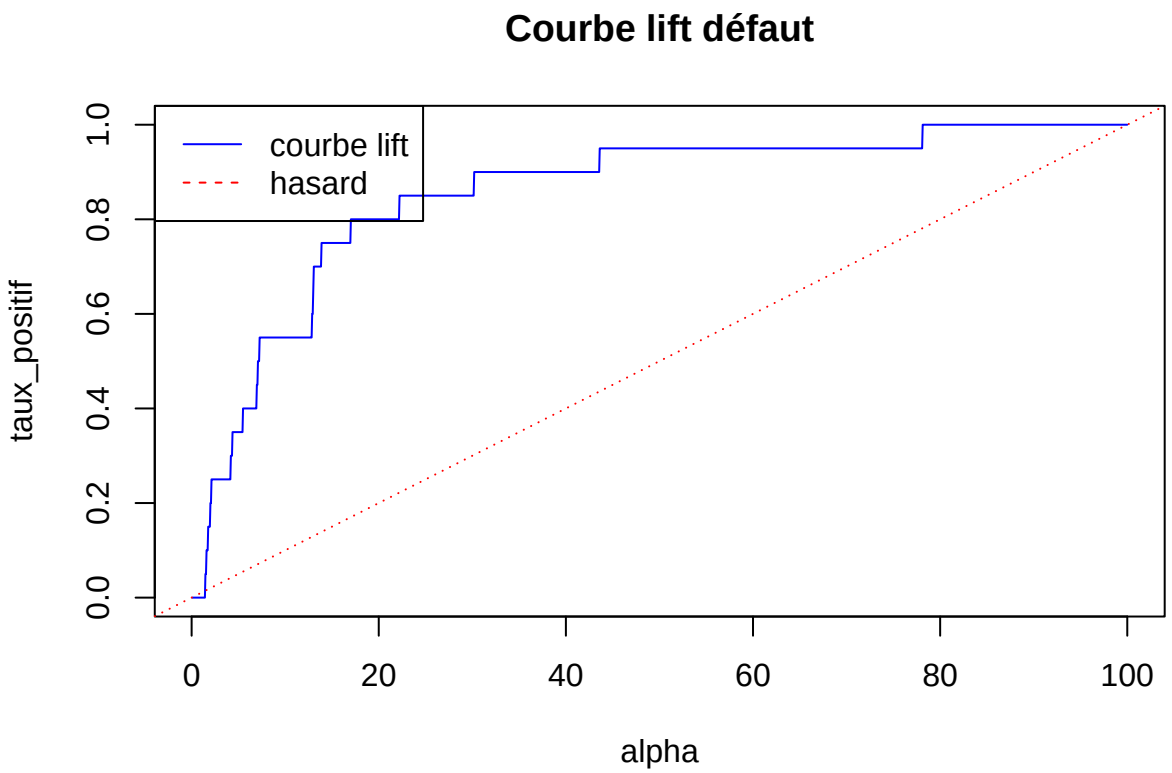
```
#tuneRF(train, train$def, mtryStart=2, ntreeTry=50, stepFactor=2, trace=TRUE, nodesize=1, type="regress
#On observe que le mtry le plus faible ave un oob de 0% est 15
```

PERFORMANCES

Courbe lift défaut On représente ici l'évolution du lift par rapport au score de défaut en fonction de l'alpha(% des données) choisi.

```
data_perf <- data_predict[order(-proba),]

taux_positif <- rep(0,nrow(data_perf))
if (data_perf[1,2]==1){
  taux_positif <- 1
}

for (i in 2:nrow(data_perf)){
  if (data_perf[i,2] ==1){
      taux_positif[i] <- taux_positif[i-1] +1
  }
  else{
    taux_positif[i] <- taux_positif[i-1]
  }
}
taux_positif <- taux_positif/ sum(data_perf[,2])

alpha <- rep(0,nrow(data_perf))
for (i in 1:nrow(data_perf)){
  alpha[i]<- i / nrow(data_perf) *100
}
```

```
plot(taux_positif~alpha, type='l', col="blue", main = "Courbe lift défaut")
abline(a=0,b=0.01, col='red', lty ='dotted')
legend('topleft', legend=c("courbe lift", "hasard"), col=c("blue","red"), lty=1:2)
```



Courbe lift octroi On représente ici l'évolution du lift par rapport au score d'octroi en fonction de l'alpha(% des données) choisi. On observe un lift equivalent au hasard, ce qui semble logique car avec un taux cible >99% (le non défaut), trier au hasard donne déja un bon score.

```
data_perf_2 <- data_predict[order(proba),]

taux_negatif <- rep(0,nrow(data_perf_2))
if (data_perf_2[1,2]==0){
  taux_negatif <- 1
}

for (i in 2:nrow(data_perf)){
  if (data_perf_2[i,2] ==0){
      taux_negatif[i] <- taux_negatif[i-1] +1
  }
  else{
    taux_negatif[i] <- taux_negatif[i-1]
  }
}
taux_negatif <- taux_negatif/ (nrow(data_perf) -sum(data_perf_2[,2]))

alpha <- rep(0,nrow(data_perf_2))
for (i in 1:nrow(data_perf_2)){
  alpha[i]<- i / nrow(data_perf_2) *100
}
```

```r
plot(taux_negatif~alpha, type='l', col="blue", main = "Courbe lift octroi")
abline(a=0,b=0.01, col='red', lty ='dotted')
legend('topleft', legend=c("courbe lift", "hasard"), col=c("blue","red"), lty=1:2)
```

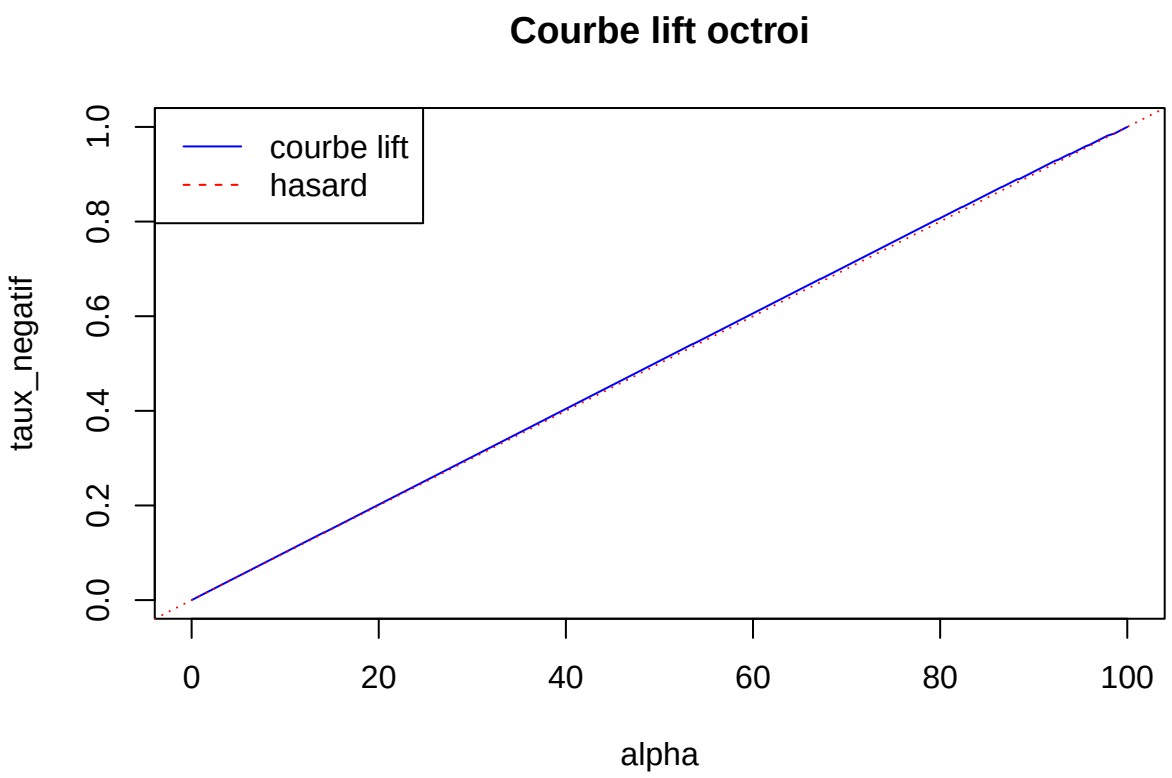


Tableau lift defaut

```r
def=test$def
tab1=order(proba,decreasing=TRUE)

i=1
s=0
j=1
k=1
nombre_positif=rep(0,10)
effectif=rep(0,10)
while (i<=1521){
  if (def[tab1[i]]==1){s=s+1}
  if (i%%169==0){
    nombre_positif[j]=s
    effectif[j]=k
    s=0
    j=j+1
    k=0

  }
  i=i+1
  k=k+1
}
```

```
effectif[10]=1692-1521
s=0
for (i in 1521:1692){
  if (def[tab1[i]]==1){s=s+1}
}
nombre_positif[10]=s

taux_cible=nombre_positif/effectif
taux_cible_total=sum(def)/length(def)

alpha=seq(10,100,10)
alpha_lift=taux_cible/taux_cible_total

alpha_lift_cumul=rep(0,10)
alpha_lift_cumul[1]=alpha_lift[1]
mem1=nombre_positif[1]
mem2=effectif[1]
for (i in 2:10){
 alpha_lift_cumul[i]=((mem1+nombre_positif[i])/(mem2+effectif[i]))*(1/taux_cible_total)
 mem1=mem1+nombre_positif[i]
 mem2=mem2+effectif[i]
}

tab5=cbind(alpha,effectif,nombre_positif,taux_cible,alpha_lift,alpha_lift_cumul)
tableau_lift=round(tab5,2)
tableau_lift
```

```
##       alpha effectif nombre_positif taux_cible alpha_lift alpha_lift_cumul
##  [1,]    10      169             11       0.07       5.51             5.51
##  [2,]    20      169              5       0.03       2.50             4.00
##  [3,]    30      169              1       0.01       0.50             2.84
##  [4,]    40      169              1       0.01       0.50             2.25
##  [5,]    50      169              1       0.01       0.50             1.90
##  [6,]    60      169              0       0.00       0.00             1.59
##  [7,]    70      169              0       0.00       0.00             1.36
##  [8,]    80      169              1       0.01       0.50             1.25
##  [9,]    90      169              0       0.00       0.00             1.11
## [10,]   100      171              0       0.00       0.00             1.00
```

Tableau lift octroi

```
def=test$def
tab1=order(proba,decreasing=F)

i=1
s=0
j=1
k=1
nombre_positif=rep(0,10)
effectif=rep(0,10)
while (i<=1521){
  if (def[tab1[i]]==0){s=s+1}
  if (i%%169==0){
    nombre_positif[j]=s
    effectif[j]=k
```

```
    s=0
    j=j+1
    k=0

  }
  i=i+1
  k=k+1
}
effectif[10]=1692-1521
s=0
for (i in 1521:1692){
  if (def[tab1[i]]==0){s=s+1}
}
nombre_positif[10]=s

taux_cible=nombre_positif/effectif
taux_cible_total=(1692-sum(def))/length(def)

alpha=seq(10,100,10)
alpha_lift=taux_cible/taux_cible_total

alpha_lift_cumul=rep(0,10)
alpha_lift_cumul[1]=alpha_lift[1]
mem1=nombre_positif[1]
mem2=effectif[1]
for (i in 2:10){
 alpha_lift_cumul[i]=((mem1+nombre_positif[i])/(mem2+effectif[i]))*(1/taux_cible_total)
 mem1=mem1+nombre_positif[i]
 mem2=mem2+effectif[i]
}

tab5=cbind(alpha,effectif,nombre_positif,taux_cible,alpha_lift,alpha_lift_cumul)
tableau_lift=round(tab5,2)
tableau_lift
```

```
##       alpha effectif nombre_positif taux_cible alpha_lift alpha_lift_cumul
##  [1,]    10      169            169       1.00       1.01             1.01
##  [2,]    20      169            168       0.99       1.01             1.01
##  [3,]    30      169            169       1.00       1.01             1.01
##  [4,]    40      169            169       1.00       1.01             1.01
##  [5,]    50      169            169       1.00       1.01             1.01
##  [6,]    60      169            168       0.99       1.01             1.01
##  [7,]    70      169            168       0.99       1.01             1.01
##  [8,]    80      169            168       0.99       1.01             1.01
##  [9,]    90      169            164       0.97       0.98             1.01
## [10,]   100      171            161       0.94       0.95             1.00
```

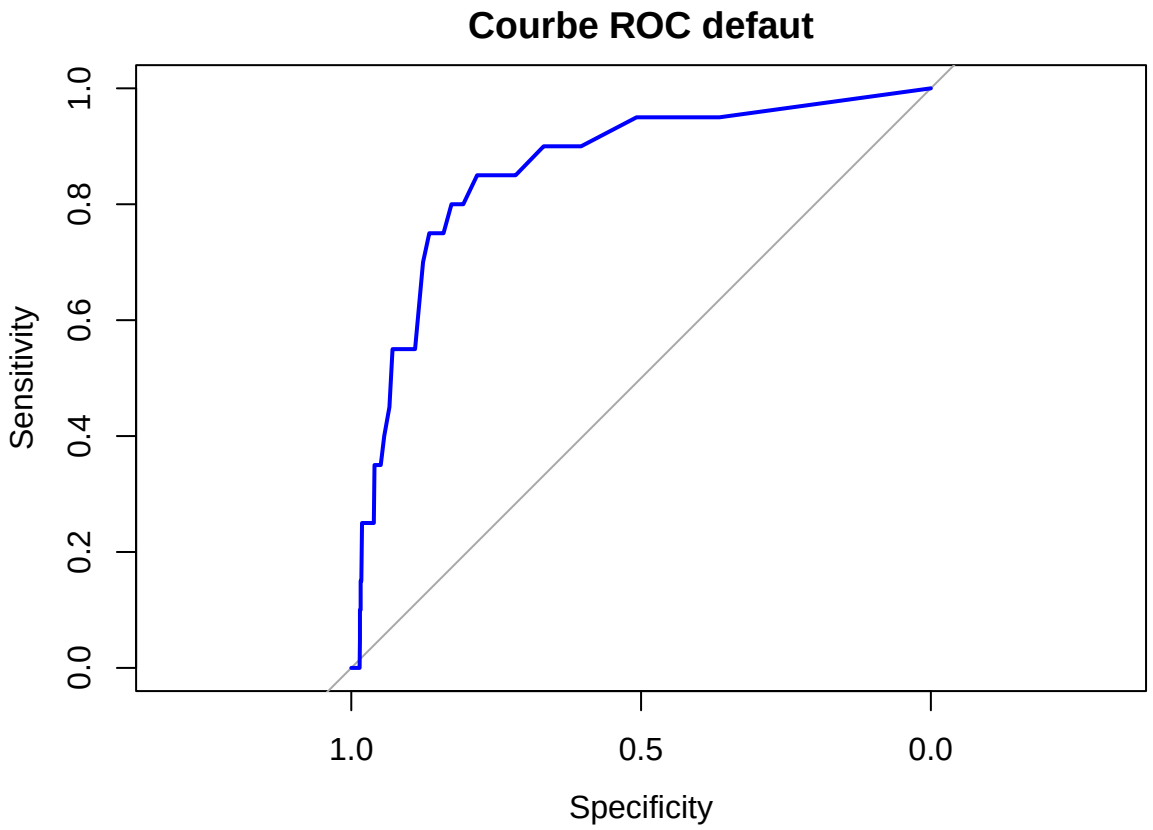
Courbes ROC et AUC On observe les courbes ROC de défaut et d'octroi, ainsi que l'AUC (aire sous la courbe ROC). AUC de 0.5 = comme le hasard AUC de 1 = score parfait

```
ROC_defaut <- roc(data_perf, response = def, predictor = proba)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(ROC_defaut, main ="Courbe ROC defaut", col="blue")
```

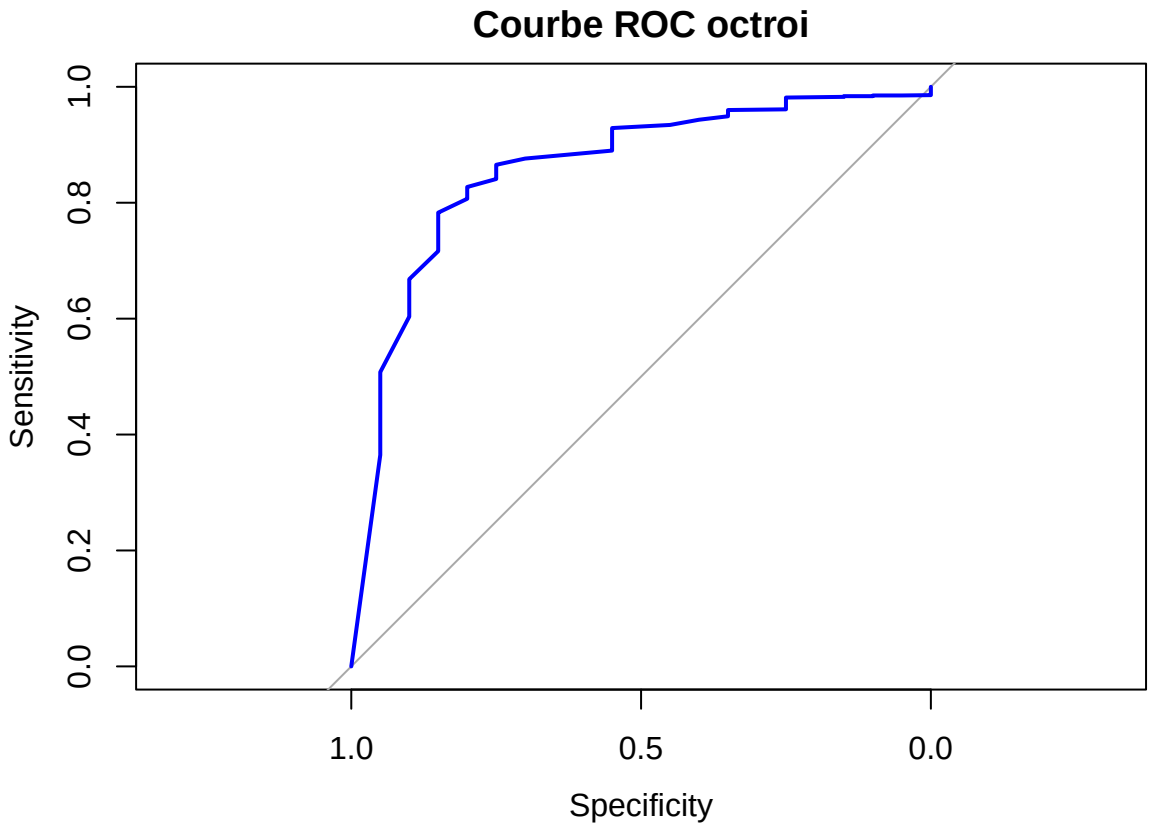



```
data_perf_3 <- data_perf_2
for (i in 1:nrow(data_perf_3)){
  if (data_perf_3[i,2]==1){
    data_perf_3[i,2]<-0
  }
  else{
    data_perf_3[i,2]<-1
  }
}
ROC_octroi <- roc(data_perf_3, response = def, predictor = proba)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
plot(ROC_octroi, main="Courbe ROC octroi", col="blue")
```




```
AUC<- auc(ROC_defaut)
print(AUC)
```

```
## Area under the curve: 0.8583
```

```
#proba de placer un 1 avant un 0

GINI <- 2*AUC-1
print(GINI)
```

```
## [1] 0.7166567
```

```
# % de mieux par rapport au hasard
```

MLP

```
remove(list=objects())
train <- read.csv2('train_non_reech.csv', sep=",",header=TRUE)
test <- read.csv2('test.csv', sep=",",header=TRUE)

train[,1]=as.numeric(train[,1])
train[,2]=as.numeric(train[,2])
train[,3]=as.numeric(train[,3])
train[,4]=as.numeric(train[,4])
train[,5]=as.numeric(train[,5])
train[,6]=as.numeric(train[,6])
train[,7]=as.numeric(train[,7])
train[,8]=as.numeric(train[,8])
```

```
train[,9]=as.numeric(train[,9])
train[,10]=as.numeric(train[,10])
train[,11]=as.numeric(train[,11])
train[,12]=as.numeric(train[,12])
train[,13]=as.numeric(train[,13])
train[,14]=as.numeric(train[,14])
train[,15]=as.numeric(train[,15])
train[,16]=as.numeric(train[,16])
train[,17]=as.numeric(train[,17])
train[,18]=as.numeric(train[,18])

test[,1]=as.numeric(test[,1])
test[,2]=as.numeric(test[,2])
test[,3]=as.numeric(test[,3])
test[,4]=as.numeric(test[,4])
test[,5]=as.numeric(test[,5])
test[,6]=as.numeric(test[,6])
test[,7]=as.numeric(test[,7])
test[,8]=as.numeric(test[,8])
test[,9]=as.numeric(test[,9])
test[,10]=as.numeric(test[,10])
test[,11]=as.numeric(test[,11])
test[,12]=as.numeric(test[,12])
test[,13]=as.numeric(test[,13])
test[,14]=as.numeric(test[,14])
test[,15]=as.numeric(test[,15])
test[,16]=as.numeric(test[,16])
test[,17]=as.numeric(test[,17])
test[,18]=as.numeric(test[,18])

xtrain <- train[,2:18]
xtrain <- normalize(xtrain)
xtrain <- as.matrix(cbind(xtrain,train[,19:81]))

xtest <- test[,2:18]
xtest <- normalize(xtest)
xtest <- as.matrix(cbind(xtest,test[,19:81]))

ytrain <- train$def
ytest <- test$def
```

MLP à deux couches cachées

architecture : une entrée avec 80 inputs 1ere couche avec 45 neurones avec fonction d'activation sigmoid 2eme couche avec 10 neurones avec fonction d'activation sigmoid une sortie avec 1 neurone avec fonction d'activation sigmoid

apprentissage : fonction de perte : binary_crossentropy algo d'optimisation : adam epoch (nbr d'apprentissage sur toutes les données) : 10 batch_size (nombre de données rentrées à la fois pour entrainer le reseau): 50

```
set_random_seed(1234)
```

```
## Loaded Tensorflow version 2.9.1
```

```
#definition du réseau
modele <- keras_model_sequential()
```

```
modele %>%
 layer_dense(units=45, input_shape =c(80), activation="sigmoid")%>%
 layer_dense(units=10,activation="sigmoid")%>%
 layer_dense(units=1,activation="sigmoid")


#infos du réseau
#print(get_config(modele))
summary(modele)
```

```
## Model: "sequential"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  dense_2 (Dense)                    (None, 45)                      3645
##  dense_1 (Dense)                    (None, 10)                      460
##  dense (Dense)                      (None, 1)                       11
## =========================================================================
## Total params: 4,116
## Trainable params: 4,116
## Non-trainable params: 0
## _________________________________________________________________________
```

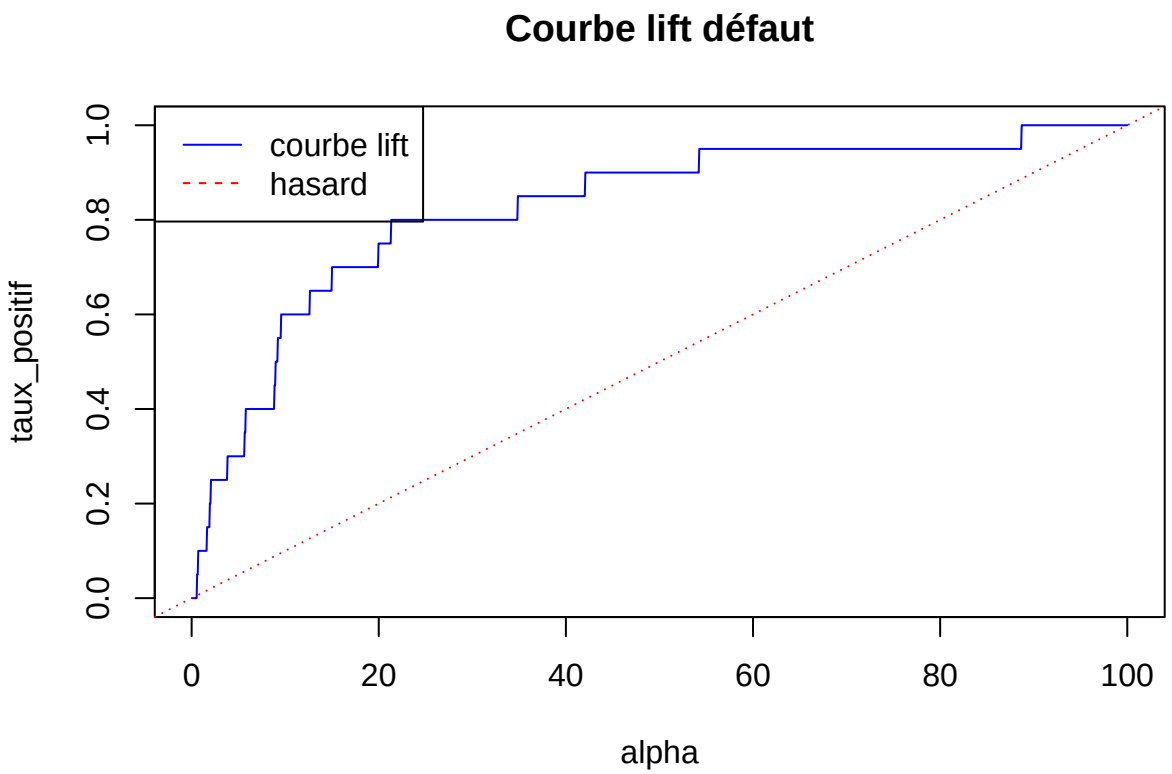
```
#compilation du réseau
modele %>% compile(
 loss="binary_crossentropy",
 optimizer="Adam"
)


#apprentissage
history <- modele %>% fit(
 x=xtrain,
 y=ytrain,
 epoch = 10,
 batch_size=50
)

proba <- modele %>% predict(xtest)
id <- as.numeric(as.character(rownames(test)))
data_predict <- as.data.frame(cbind(proba,test[1]))

scoring <- as.data.frame(cbind(proba, test[1]))
write.csv(scoring, "scoring_MLP_non_reech.csv")
```

PERFORMANCES

Courbe lift défaut On représente ici l'évolution du lift par rapport au score de défaut en fonction de l'alpha(% des données) choisi.

```
data_perf <- data_predict[order(-proba),]

taux_positif <- rep(0,nrow(data_perf))
if (data_perf[1,2]==1){
  taux_positif <- 1
```

```
}

for (i in 2:nrow(data_perf)){
  if (data_perf[i,2] ==1){
      taux_positif[i] <- taux_positif[i-1] +1
  }
  else{
    taux_positif[i] <- taux_positif[i-1]
  }
}
taux_positif <- taux_positif/ sum(data_perf[,2])

alpha <- rep(0,nrow(data_perf))
for (i in 1:nrow(data_perf)){
  alpha[i]<- i / nrow(data_perf) *100
}

plot(taux_positif~alpha, type='l', col="blue", main = "Courbe lift défaut")
abline(a=0,b=0.01, col='red', lty ='dotted')
legend('topleft', legend=c("courbe lift", "hasard"), col=c("blue","red"), lty=1:2)
```



Courbe lift octroi On représente ici l'évolution du lift par rapport au score d'octroi en fonction de l'alpha(% des données) choisi. On observe un lift equivalent au hasard, ce qui semble logique car avec un taux cible >99% (le non défaut), trier au hasard donne déja un bon score.

```
data_perf_2 <- data_predict[order(proba),]

taux_negatif <- rep(0,nrow(data_perf_2))
if (data_perf_2[1,2]==0){
  taux_negatif <- 1
```

```
}

for (i in 2:nrow(data_perf)){
  if (data_perf_2[i,2] ==0){
      taux_negatif[i] <- taux_negatif[i-1] +1
  }
  else{
    taux_negatif[i] <- taux_negatif[i-1]
  }
}
taux_negatif <- taux_negatif/ (nrow(data_perf) -sum(data_perf_2[,2]))

alpha <- rep(0,nrow(data_perf_2))
for (i in 1:nrow(data_perf_2)){
  alpha[i]<- i / nrow(data_perf_2) *100
}


plot(taux_negatif~alpha, type='l', col="blue", main = "Courbe lift octroi")
abline(a=0,b=0.01, col='red', lty ='dotted')
legend('topleft', legend=c("courbe lift", "hasard"), col=c("blue","red"), lty=1:2)
```

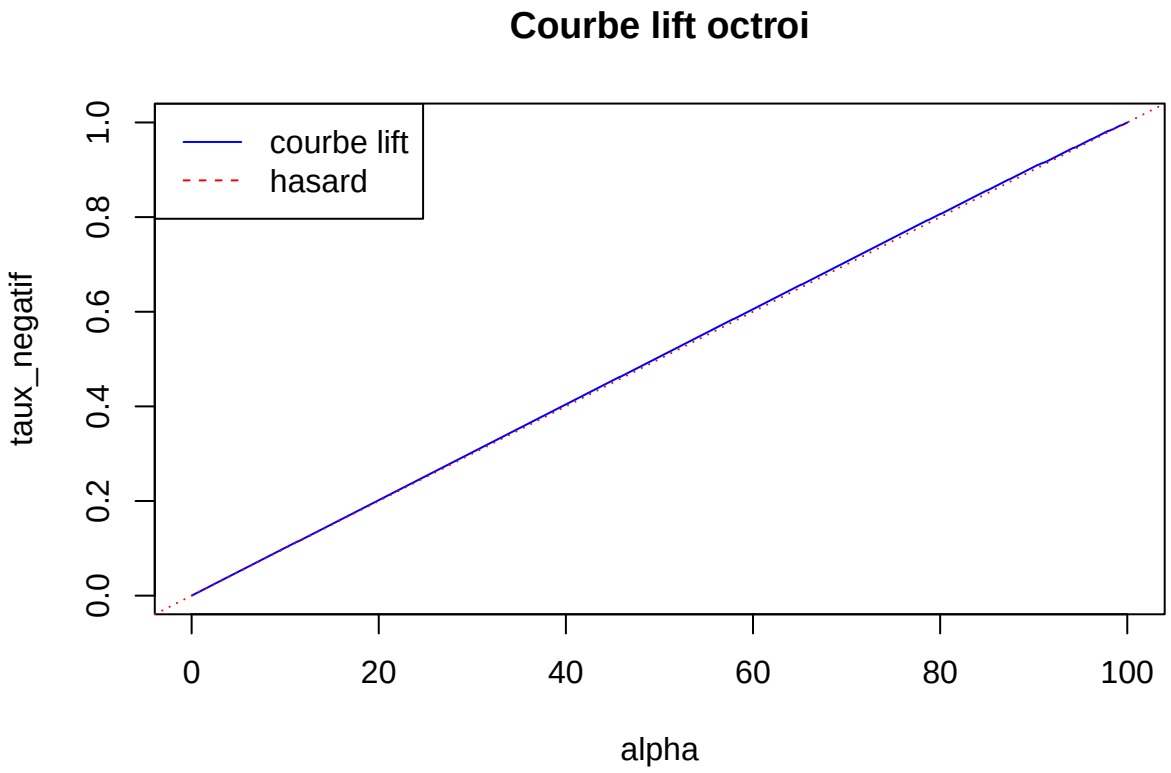


Tableau lift defaut

```
def=test$def
tab1=order(proba,decreasing=TRUE)

i=1
s=0
j=1
```

```
k=1
nombre_positif=rep(0,10)
effectif=rep(0,10)
while (i<=1521){
  if (def[tab1[i]]==1){s=s+1}
  if (i%%169==0){
    nombre_positif[j]=s
    effectif[j]=k
    s=0
    j=j+1
    k=0

  }
  i=i+1
  k=k+1
}
effectif[10]=1692-1521
s=0
for (i in 1521:1692){
  if (def[tab1[i]]==1){s=s+1}
}
nombre_positif[10]=s

taux_cible=nombre_positif/effectif
taux_cible_total=sum(def)/length(def)

alpha=seq(10,100,10)
alpha_lift=taux_cible/taux_cible_total

alpha_lift_cumul=rep(0,10)
alpha_lift_cumul[1]=alpha_lift[1]
mem1=nombre_positif[1]
mem2=effectif[1]
for (i in 2:10){
 alpha_lift_cumul[i]=((mem1+nombre_positif[i])/(mem2+effectif[i]))*(1/taux_cible_total)
 mem1=mem1+nombre_positif[i]
 mem2=mem2+effectif[i]
}

tab5=cbind(alpha,effectif,nombre_positif,taux_cible,alpha_lift,alpha_lift_cumul)
tableau_lift=round(tab5,2)
tableau_lift
```

```
##       alpha effectif nombre_positif taux_cible alpha_lift alpha_lift_cumul
##  [1,]    10      169             12       0.07       6.01             6.01
##  [2,]    20      169              3       0.02       1.50             3.75
##  [3,]    30      169              1       0.01       0.50             2.67
##  [4,]    40      169              1       0.01       0.50             2.13
##  [5,]    50      169              1       0.01       0.50             1.80
##  [6,]    60      169              1       0.01       0.50             1.59
##  [7,]    70      169              0       0.00       0.00             1.36
##  [8,]    80      169              0       0.00       0.00             1.19
##  [9,]    90      169              1       0.01       0.50             1.11
## [10,]   100      171              0       0.00       0.00             1.00
```

Tableau lift octroi

```
def=test$def
tab1=order(proba,decreasing=F)

i=1
s=0
j=1
k=1
nombre_positif=rep(0,10)
effectif=rep(0,10)
while (i<=1521){
  if (def[tab1[i]]==0){s=s+1}
  if (i%%169==0){
    nombre_positif[j]=s
    effectif[j]=k
    s=0
    j=j+1
    k=0

  }
  i=i+1
  k=k+1
}
effectif[10]=1692-1521
s=0
for (i in 1521:1692){
  if (def[tab1[i]]==0){s=s+1}
}
nombre_positif[10]=s

taux_cible=nombre_positif/effectif
taux_cible_total=(1692-sum(def))/length(def)

alpha=seq(10,100,10)
alpha_lift=taux_cible/taux_cible_total

alpha_lift_cumul=rep(0,10)
alpha_lift_cumul[1]=alpha_lift[1]
mem1=nombre_positif[1]
mem2=effectif[1]
for (i in 2:10){
 alpha_lift_cumul[i]=((mem1+nombre_positif[i])/(mem2+effectif[i]))*(1/taux_cible_total)
 mem1=mem1+nombre_positif[i]
 mem2=mem2+effectif[i]
}

tab5=cbind(alpha,effectif,nombre_positif,taux_cible,alpha_lift,alpha_lift_cumul)
tableau_lift=round(tab5,2)
tableau_lift
```

```
##       alpha effectif nombre_positif taux_cible alpha_lift alpha_lift_cumul
##  [1,]    10      169            169       1.00       1.01             1.01
##  [2,]    20      169            168       0.99       1.01             1.01
##  [3,]    30      169            169       1.00       1.01             1.01
```

```
##  [4,]    40      169             169       1.00       1.01             1.01
##  [5,]    50      169             168       0.99       1.01             1.01
##  [6,]    60      169             168       0.99       1.01             1.01
##  [7,]    70      169             168       0.99       1.01             1.01
##  [8,]    80      169             168       0.99       1.01             1.01
##  [9,]    90      169             166       0.98       0.99             1.01
## [10,]   100      171             160       0.94       0.95             1.00
```

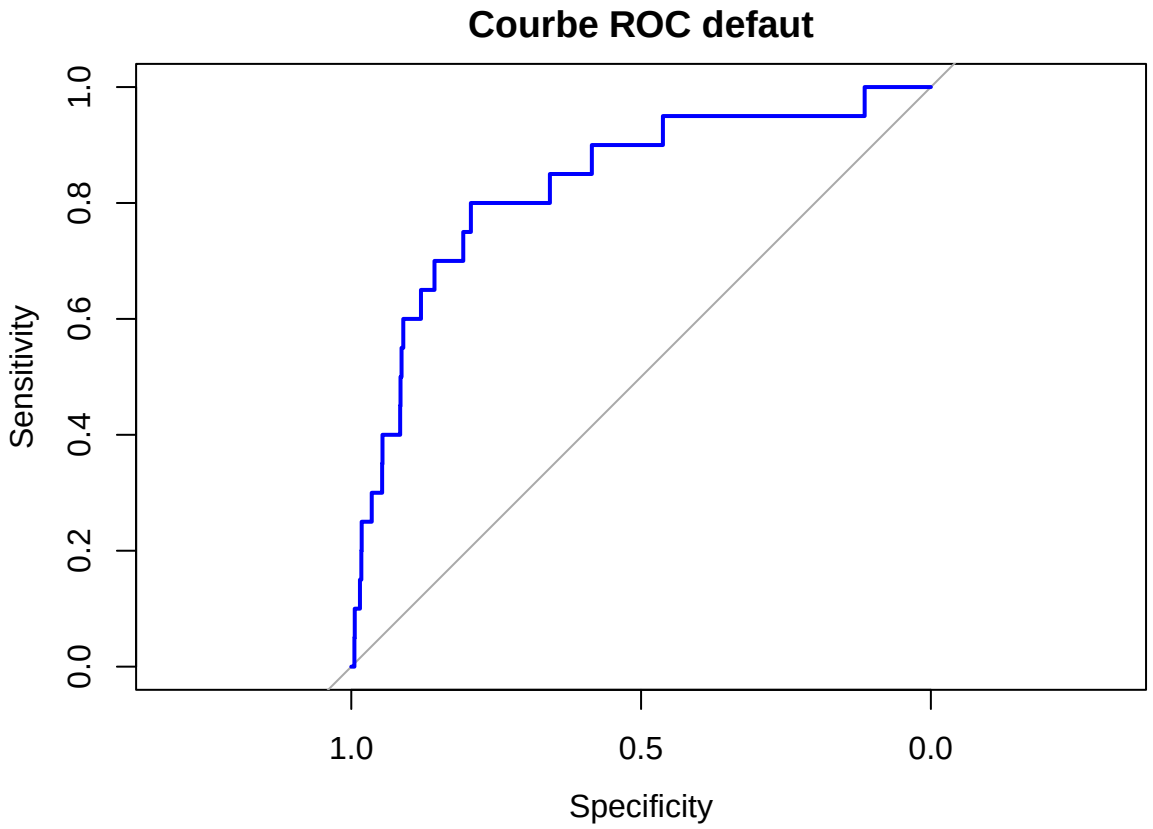
Courbes ROC et AUC On observe les courbes ROC de défaut et d'octroi, ainsi que l'AUC (aire sous la courbe ROC). AUC de 0.5 = comme le hasard AUC de 1 = score parfait

```
ROC_defaut <- roc(data_perf, response = def, predictor = proba)
```

```
## Setting levels: control = 0, case = 1
```

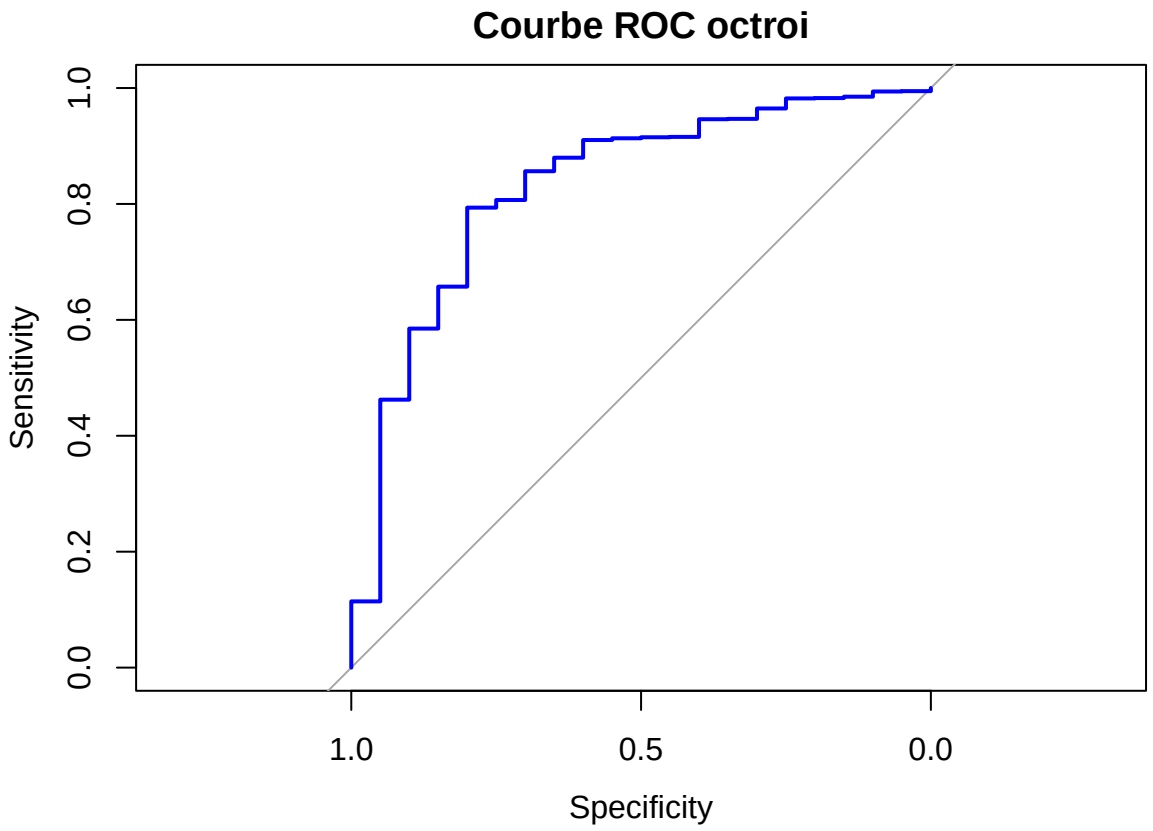
```
## Setting direction: controls < cases
```

```
plot(ROC_defaut, main ="Courbe ROC defaut", col="blue")
```




```
data_perf_3 <- data_perf_2
for (i in 1:nrow(data_perf_3)){
  if (data_perf_3[i,2]==1){
    data_perf_3[i,2]<-0
  }
  else{
    data_perf_3[i,2]<-1
  }
}
ROC_octroi <- roc(data_perf_3, response = def, predictor = proba)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls > cases
plot(ROC_octroi, main="Courbe ROC octroi", col="blue")
```




```
AUC<- auc(ROC_defaut)
print(AUC)
```

```
## Area under the curve: 0.8303
```

```
#proba de placer un 1 avant un 0

GINI <- 2*AUC-1
print(GINI)
```

```
## [1] 0.6605861
```

```
# % de mieux par rapport au hasard
```