Application des methodes d'apprentissage automatique dans l'étude de la survie des enfants au Burkina-Faso

Adjiwanou - Aoudou

24 June 2024

```
rm(list = ls())
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(summarytools)
```

```
##
## Attaching package: 'summarytools'
##
## The following object is masked from 'package:tibble':
##
##     view
```

```
library(gtsummary)
library(haven)
```

Chargement des données

```
load("base_bf_complete.rda")
```

Analyse descriptive

Comme d'habitude, avant tout, il faut toujours commencer par décrire la base de données afin de comprendre le jeu de données et de mener des analyses descriptives préliminaires afin d'avoir une première idée du sujet

mis à l'étude. Ici, la variable d'intérêt principale est l'état de survie de l'enfant; qui est une variable binaire dont l'objectif est la construction des modèles de classification pouvant mieux discriminer ses modalités.

Regardons les associations entre la variable à prédire **dead** avec toutes les prédicteurs considérés.

```
data <- base %>% select(-m19,-v012)

data %>%
  tbl_summary(
    include = c(educ,activite,attitude_violence,pouvoir_decision,age_mere,degmedia,ins_conj,sex_enfant,
    by = dead
  ) %>%
  add_p(
    test = all_categorical() ~ "chisq.test",
    pvalue_fun = scales::label_pvalue(accuracy = .001, decimal.mark = ",")
  )
```

```
## Table printed with `knitr::kable()`, not {gt}. Learn why at
## https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
## To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	**a live**, N = 11,853	**dead**, N = 490	**p-value**
educ			0,002
Sans instruction	8,208 (69%)	366 (75%)	
Primaire	1,623 (14%)	70 (14%)	
Secondaire	1,881 (16%)	54 (11%)	
Supérieur	141 (1.2%)	0 (0%)	
activite			0,774
Entrepreneures agricoles	4,382 (37%)	188 (38%)	
Travailleuses qualifiées ou non qualifiées	3,363 (28%)	139 (28%)	
Sans emploi	4,108 (35%)	163 (33%)	
attitude_violence			0,593
Non favorable	6,155 (52%)	261 (53%)	
Favorable	5,698 (48%)	229 (47%)	
pouvoir_decision			0,441
Elévé	326 (2.8%)	13 (2.7%)	
Moyen	2,011 (17%)	94 (19%)	
Faible	9,516 (80%)	383 (78%)	
age_mere			<0,001
Moins de 20 ans	600 (5.1%)	29 (5.9%)	
Entre 20 et 29 ans	5,593 (47%)	183 (37%)	
Entre 30 ans et 39 ans	4,510 (38%)	189 (39%)	
40 ans et plus	1,150 (9.7%)	89 (18%)	
degmedia			0,024
Nul	3,805 (32%)	177 (36%)	
Faible	5,043 (43%)	213 (43%)	
moyenne	2,776 (23%)	97 (20%)	
Elevé	229 (1.9%)	3 (0.6%)	
ins_conj			0,008
Sans instruction	8,637 (73%)	379 (77%)	
Primaire	1,495 (13%)	67 (14%)	

Characteristic	**a live**, N = 11,853	**dead**, N = 490	**p-value**
Secondaire	1,387 (12%)	36 (7.3%)	
Supérieur	334 (2.8%)	8 (1.6%)	
sex_enfant			<0,001
Masculin	5,971 (50%)	290 (59%)	
Feminin	5,882 (50%)	200 (41%)	
poids_nais			<0,001
Faible	1,201 (10%)	126 (26%)	
Normal	10,319 (87%)	348 (71%)	
Elevé	333 (2.8%)	16 (3.3%)	
rang_naiss			<0,001
Premier né	2,567 (22%)	92 (19%)	
Rang 2 ou 3	4,281 (36%)	134 (27%)	
Rang 4 ou plus	5,005 (42%)	264 (54%)	
interval_precedent			<0,001
Moins de 24 mois	899 (7.6%)	101 (21%)	
Plus de 24 mois	8,359 (71%)	294 (60%)	
Non concerné	2,595 (22%)	95 (19%)	
lieu_accouch			0,003
Domicile	541 (4.6%)	37 (7.6%)	
Formation sanitaire	11,312 (95%)	453 (92%)	
naissance_voulu			0,099
avait voulu	10,489 (88%)	448 (91%)	
voulu pour plutard	1,168 (9.9%)	38 (7.8%)	
indésirée	196 (1.7%)	4 (0.8%)	
allaiter_heure			0,881
Oui	7,980 (67%)	332 (68%)	
Non	3,873 (33%)	158 (32%)	
source_eau			0,013
Amelioree	1,778 (15%)	53 (11%)	
Non amelioree	10,075 (85%)	437 (89%)	
type_toilet			0,050
Amelioree	6,974 (59%)	266 (54%)	
Non amelioree	4,879 (41%)	224 (46%)	
contraception			0,002
Non	7,108 (60%)	329 (67%)	
Oui	4,745 (40%)	161 (33%)	
taille_menage			0,007
2-3	759 (6.4%)	49 (10%)	
4-6	3,849 (32%)	156 (32%)	
7 et plus	7,245 (61%)	285 (58%)	
sex_chef			0,031
Masculin	10,834 (91%)	462 (94%)	
Feminin	1,019 (8.6%)	28 (5.7%)	
niveau_vie			<0,001
Pauvre	4,652 (39%)	236 (48%)	
Moyen	2,623 (22%)	113 (23%)	
Riche	4,578 (39%)	141 (29%)	
milieu_residence			<0,001
Urbain	3,370 (28%)	101 (21%)	
Rural	8,483 (72%)	389 (79%)	

Estimation du modèle de regression logistique

- Division de la base de donnée train et test

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
set.seed(123)
training_samples <- data$dead %>%
  createDataPartition(p = 0.8, list = FALSE)


data_train <- data[training_samples, ]
data_test<- data[-training_samples, ]

? createDataPartition

freq(data_train$dead)
```

```
## Frequencies
## data_train$dead
## Type: Factor
##
##                Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
## ------------ ------ --------- -------------- --------- --------------
##       a live   9483     96.03          96.03     96.03          96.03
##         dead    392      3.97         100.00      3.97         100.00
##         <NA>      0                                0.00         100.00
##        Total   9875    100.00         100.00    100.00         100.00
```

```
freq(data_test$dead)
```

```
## Frequencies
## data_test$dead
## Type: Factor
##
##                Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
## ------------ ------ --------- -------------- --------- --------------
##       a live   2370     96.03          96.03     96.03          96.03
##         dead     98      3.97         100.00      3.97         100.00
##         <NA>      0                                0.00         100.00
##        Total   2468    100.00         100.00    100.00         100.00
```

Dans le data train, il y a un gros problème de déséquilibre de données. Construire un modèle sur de telles données aura tendance à privilégier le classement de la classe majoritaire. Dans ces conditions, il faut rééquilibrer les données. Pour cela il existe plusieurs technique d'équilibrage de donnée. Soit on suréchantillonne la classe minoritaire en augmentant de façon synthétique le nombre d'exemples de la classe minoritaire, soit en sous-échantillonnant la classe majoritaire. Dans le cadre de cet exemple, on va suréchantillonner la classe minoritaire par simple réplication aléatoire

```
majoritaire <- filter(data_train, dead == "a live")
minoritaire <- filter(data_train, dead == "dead")

difference_taille <- nrow(majoritaire) - nrow(minoritaire)
difference_taille
```

```
## [1] 9091
```

On va dupliquer de façon aléatoire des échantillons de la classe minoritaire jusqu'à atteindre un équilibre avec la classe majoritaire.

```
echantillon_duplique <- sample_n(minoritaire, difference_taille, replace = TRUE)

freq(echantillon_duplique$dead)
```

```
## Frequencies
## echantillon_duplique$dead
## Type: Factor
##
##                Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
## ------------ ------ --------- -------------- --------- --------------
##       a live      0      0.00           0.00      0.00           0.00
##         dead   9091    100.00         100.00    100.00         100.00
##         <NA>      0                                0.00         100.00
##        Total   9091    100.00         100.00    100.00         100.00
```

```
# Mettre une variable qui indique que cela provient d'un échantillon dupliqué
```

Combinaison des données dupliquées avec la classe majoritaire pour former le nouvel ensemble de données équilibré

```
data_train_bal2 <- rbind(majoritaire, echantillon_duplique)
freq(data_train_bal2$dead)
```

```
## Frequencies
## data_train_bal2$dead
## Type: Factor
##
##                Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
## ------------ ------- --------- -------------- --------- --------------
##       a live    9483     51.06          51.06     51.06          51.06
##         dead    9091     48.94         100.00     48.94         100.00
##         <NA>       0                                0.00         100.00
##        Total   18574    100.00         100.00    100.00         100.00
```

Remarque : Il convient de noter que la duplication aléatoire peut introduire un certain degré de surapprentissage (overfitting), car les exemples dupliqués sont essentiellement des répétitions des exemples existants. Il est recommandé de tester différentes techniques de gestion du déséquilibre des données et de choisir celle qui convient le mieux à votre jeu de données spécifique.

- construction du modèle

```
model_logit <- glm(dead ~ ., data = data_train_bal2, family = binomial)

model_logit %>% summary()
```

```
##
## Call:
## glm(formula = dead ~ ., family = binomial, data = data_train_bal2)
##
## Coefficients:
##                                                  Estimate Std. Error z value
## (Intercept)                                       2.45741    0.32567   7.546
## educPrimaire                                      0.18507    0.05014   3.691
## educSecondaire                                   -0.09473    0.05806  -1.632
## educSupérieur                                   -14.50369   75.30598  -0.193
## activiteTravailleuses qualifiées ou non qualifiées  0.23897    0.04420   5.406
## activiteSans emploi                              -0.11541    0.03944  -2.927
## attitude_violenceFavorable                       -0.13277    0.03246  -4.090
## pouvoir_decisionMoyen                             0.10712    0.10144   1.056
## pouvoir_decisionFaible                           -0.02836    0.09599  -0.295
## age_mereEntre 20 et 29 ans                        0.22423    0.08144   2.754
## age_mereEntre 30 ans et 39 ans                    0.24378    0.09329   2.613
## age_mere40 ans et plus                            0.90388    0.10342   8.740
## degmediaFaible                                   -0.03155    0.03726  -0.847
## degmediamoyenne                                  -0.12065    0.04953  -2.436
## degmediaElevé                                    -0.58865    0.18407  -3.198
## ins_conjPrimaire                                 -0.05352    0.05035  -1.063
## ins_conjSecondaire                               -0.13648    0.06362  -2.145
## ins_conjSupérieur                                 0.43823    0.13048   3.359
## sex_enfantFeminin                                -0.31403    0.03207  -9.792
## poids_naisNormal                                 -1.18266    0.04445 -26.607
## poids_naisElevé                                  -0.85234    0.10254  -8.312
## rang_naissRang 2 ou 3                             0.64093    0.27419   2.337
## rang_naissRang 4 ou plus                          0.97537    0.28222   3.456
## interval_precedentPlus de 24 mois                -1.12901    0.05120 -22.050
## interval_precedentNon concerné                   -0.18430    0.27709  -0.665
## lieu_accouchFormation sanitaire                  -0.48758    0.06856  -7.112
## naissance_vouluvoulu pour plutard                -0.38335    0.05927  -6.468
## naissance_vouluindésirée                         -1.24471    0.16045  -7.758
## allaiter_heureNon                                 0.11005    0.03441   3.198
## source_eauNon amelioree                          -0.27545    0.05809  -4.741
## type_toiletNon amelioree                         -0.02188    0.03796  -0.577
## contraceptionOui                                 -0.29474    0.03445  -8.555
## taille_menage4-6                                 -0.70761    0.06513 -10.864
## taille_menage7 et plus                           -0.84810    0.06372 -13.311
## sex_chefFeminin                                  -0.60234    0.06801  -8.856
## niveau_vieMoyen                                  -0.02720    0.04315  -0.630
```

```
## niveau_vieRiche                                      -0.24151    0.05059  -4.774
## milieu_residenceRural                                 0.26207    0.05078   5.161
##                                                    Pr(>|z|)
## (Intercept)                                        4.50e-14 ***
## educPrimaire                                       0.000224 ***
## educSecondaire                                     0.102778
## educSupérieur                                      0.847275
## activiteTravailleuses qualifiées ou non qualifiées 6.44e-08 ***
## activiteSans emploi                                0.003428 **
## attitude_violenceFavorable                         4.32e-05 ***
## pouvoir_decisionMoyen                              0.290954
## pouvoir_decisionFaible                             0.767619
## age_mereEntre 20 et 29 ans                         0.005896 **
## age_mereEntre 30 ans et 39 ans                     0.008975 **
## age_mere40 ans et plus                              < 2e-16 ***
## degmediaFaible                                     0.397154
## degmediamoyenne                                    0.014850 *
## degmediaElevé                                      0.001384 **
## ins_conjPrimaire                                   0.287843
## ins_conjSecondaire                                 0.031933 *
## ins_conjSupérieur                                  0.000784 ***
## sex_enfantFeminin                                   < 2e-16 ***
## poids_naisNormal                                    < 2e-16 ***
## poids_naisElevé                                     < 2e-16 ***
## rang_naissRang 2 ou 3                              0.019413 *
## rang_naissRang 4 ou plus                           0.000548 ***
## interval_precedentPlus de 24 mois                   < 2e-16 ***
## interval_precedentNon concerné                     0.505971
## lieu_accouchFormation sanitaire                    1.15e-12 ***
## naissance_vouluvoulu pour plutard                  9.93e-11 ***
## naissance_vouluindésirée                           8.65e-15 ***
## allaiter_heureNon                                  0.001383 **
## source_eauNon amelioree                            2.12e-06 ***
## type_toiletNon amelioree                           0.564275
## contraceptionOui                                    < 2e-16 ***
## taille_menage4-6                                    < 2e-16 ***
## taille_menage7 et plus                              < 2e-16 ***
## sex_chefFeminin                                     < 2e-16 ***
## niveau_vieMoyen                                    0.528429
## niveau_vieRiche                                    1.81e-06 ***
## milieu_residenceRural                              2.46e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25741  on 18573  degrees of freedom
## Residual deviance: 23011  on 18536  degrees of freedom
## AIC: 23087
##
## Number of Fisher Scoring iterations: 13
```

- prediction sur le train et le test set

```r
# sur le data train

prediction_train <- predict(model_logit, newdata = data_train_bal2,type = "response")

predicted_class_train <- if_else(prediction_train > 0.5, "dead", "alive")

# sur le test set

y <- lapply(data_test, as.factor)
y <- as.data.frame(y)
data_test = y

predicted_test <- predict(model_logit, newdata = data_test, type = "response")
predicted_class_test <- if_else(predicted_test > 0.5, "dead", "alive")
```

- Validation du modèle: Il existe plusieurs métriques de validation; nous présentons quelques unes.

```r
evaluation_prediction <- function(yobs, ypred, posLabel = 1) {

    # Matrice de confusion
  mc <- table(yobs, ypred)

  # Taux de bon classement (Accuracy)
  tbc <- round(sum(diag(mc)) / sum(mc), 4)

  # Rappel / recall
  recall <- mc[posLabel, posLabel] / sum(mc[posLabel, ])

  # Précision
  precision <- mc[posLabel, posLabel] / sum(mc[, posLabel])

  # F1-Measure

  f1 <- 2.0 * (precision * recall) / (precision + recall)

  # Créer le tableau des métriques

  metrics <- data.frame(
    Taux_de_bon_classement = tbc,
    Rappel = round(recall, 3),
    Précision = round(precision, 3),
    F1_Score = round(f1, 3)
  )

  # Retourner le tableau des métriques
  return(metrics)
}
```

```r
evaluation_prediction(data_train_bal2$dead, predicted_class_train)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.6691  0.718     0.662    0.689
```

```
evaluation_prediction(data_test$dead, predicted_class_test)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                  0.718  0.722     0.979    0.831
```

- Pourquoi la précision est meilleure ici?
- Faire différemment en sous-échantillonnant la classe majoritaire

Regularisation

Regression logistique pénalisée LASSO

- création de la matrice des prédicteurs (sous forme exploitable par l'algorithme) comme on la fait dans le cas linéaire

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-8
```

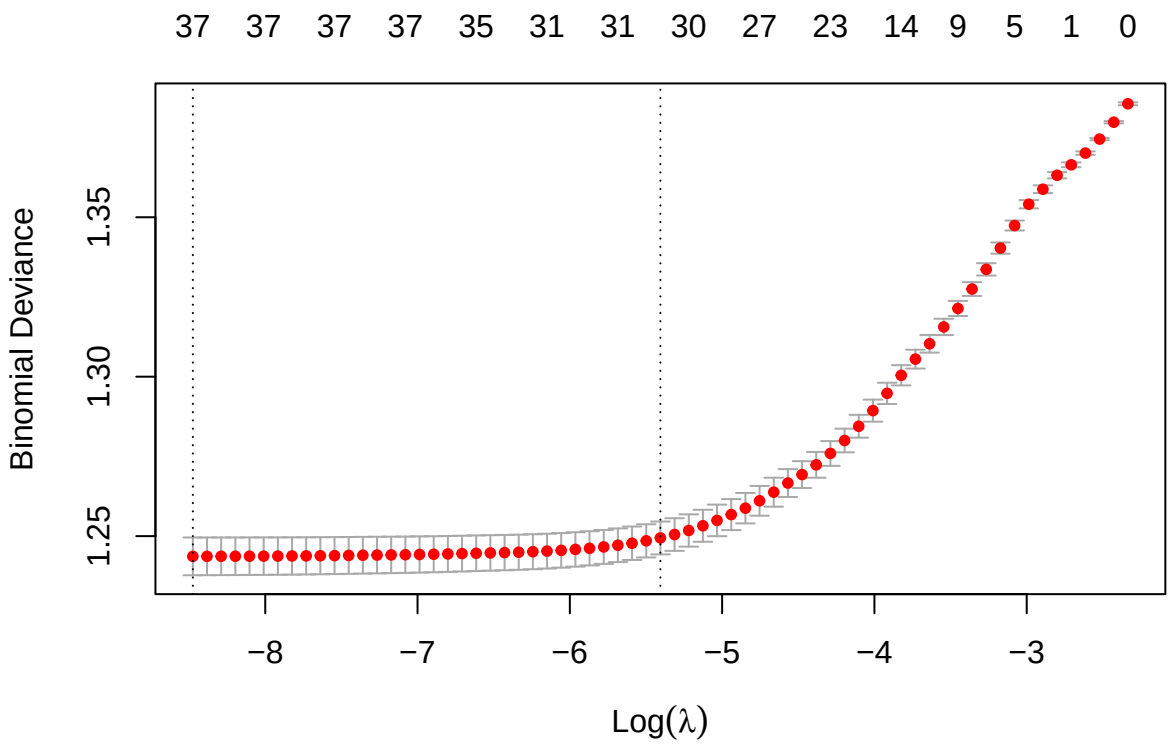
```
x_train <- model.matrix(dead~., data_train_bal2)[,-1]
y_train <- data_train_bal2$dead
```

Recherche du meilleur lambda par cross-validation à 10 plis

```
cv_lasso <- cv.glmnet(x_train, y_train, alpha =1, nfolds =10 , intercept= TRUE, family = "binomial", sta
```

Visualisation des résultats de la validation croisée avec régularisation Lasso (la valeur optimale de lambda qui minimise l'erreur de validation croisée est en pointillé)

```
plot(cv_lasso)
```



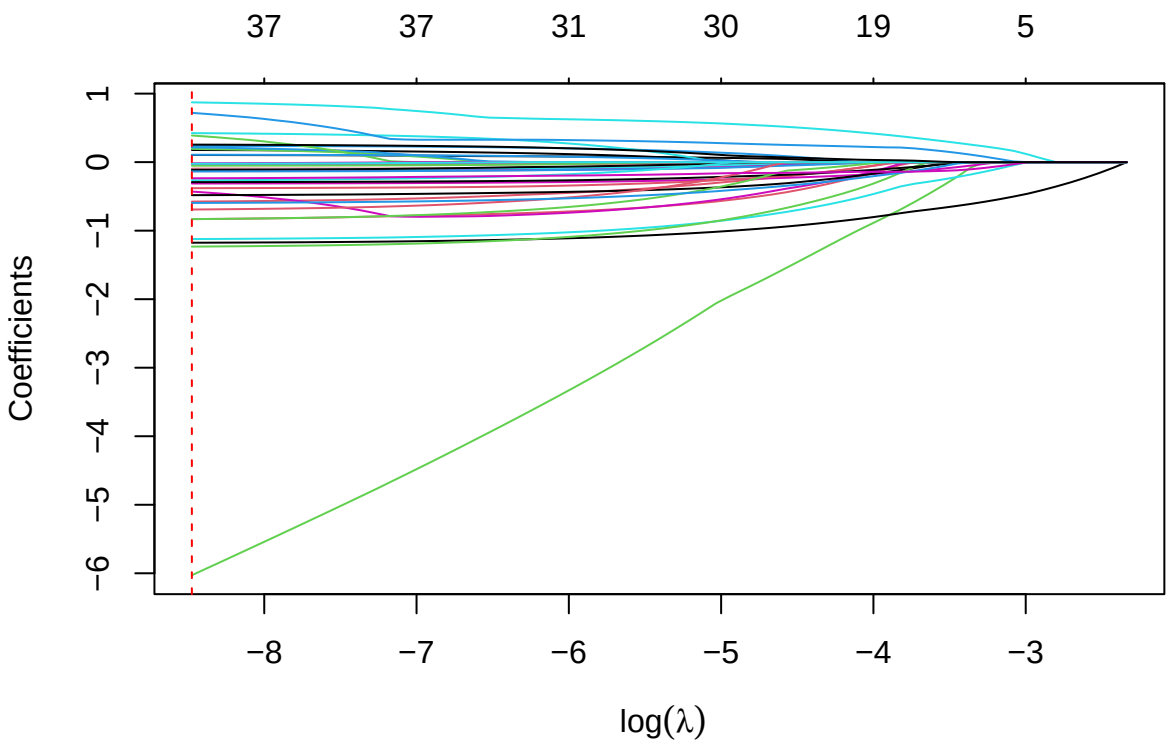
Le graphique affiche l'erreur de validation croisée en fonction du logarithme de lambda. La ligne verticale pointillée de gauche indique que le logarithme de la valeur optimale de lambda est d'environ exp(-5.4), ce qui est celui qui minimise l'erreur de prédiction. Cette valeur de lambda donnera le modèle le plus précis. La valeur exacte de lambda peut être visualisée comme suit :

```
cv_lasso$lambda.min
```

```
## [1] 0.0002085349
```

Entrainement du modèle LASSO, puis recherche du LASSO optimal

```
model_lasso <- glmnet(x_train,y_train, alpha = 1, family = "binomial")

#Visualisation de l'évolution des coefficients selon valeur de lambda avec régularisation LASSO + ligne

plot(model_lasso, xvar = "lambda", label = FALSE, xlab = ~ log(lambda))
abline( v = log(cv_lasso$lambda.min), col = "red", lty = 2)
```



- Modèle Lasso optimal

```
model_lasso <- glmnet(x_train, y_train, alpha = 1, lambda=cv_lasso$lambda.min, family = "binomial")
```

- Prédiction

```
pred_lasso_train=predict(model_lasso,newx = x_train,type = "response")
pred_lasso_train_class=if_else(pred_lasso_train >0.5,"dead","alive")


x_test <- model.matrix(dead~., data_test)[,-1]
pred_lasso=predict(model_lasso,newx = x_test,type = "response")

pred_lasso_class=if_else(pred_lasso >0.5,"dead","alive")
```

- Evaluation des performances

```
evaluation_prediction(data_train_bal2$dead,pred_lasso_train_class)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.6682  0.719     0.661    0.689
```

```
evaluation_prediction(data_test$dead,pred_lasso_class)
```

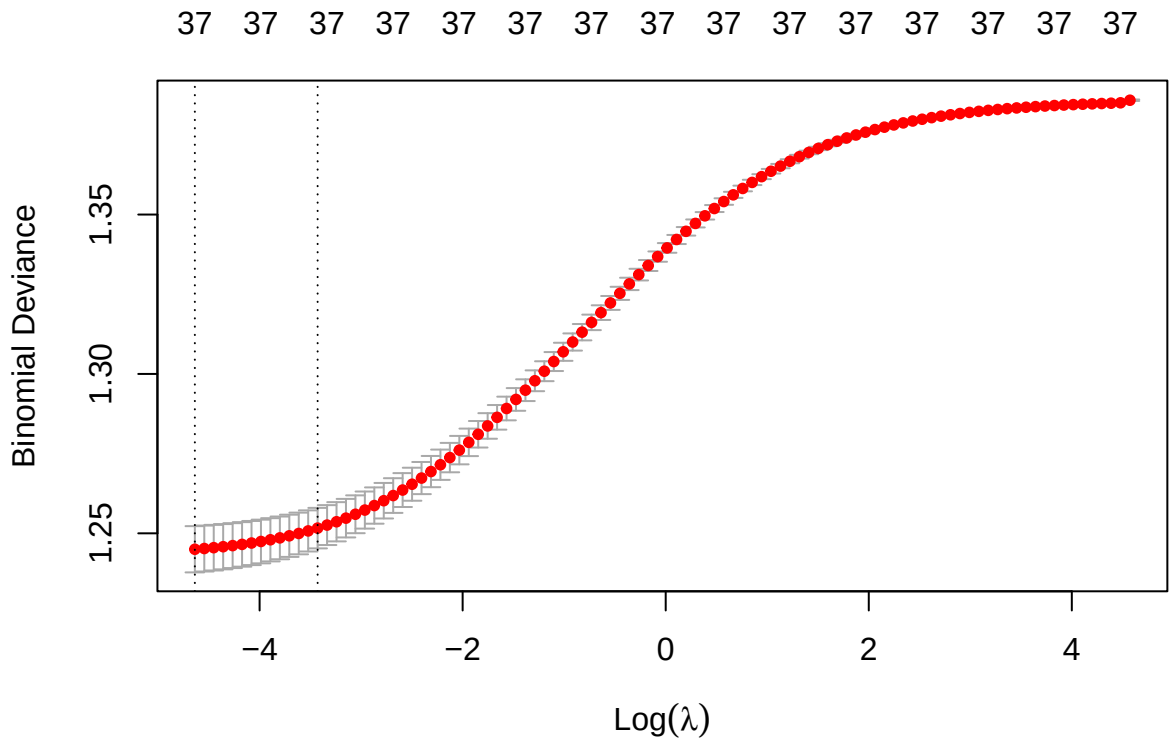
```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.7188  0.723     0.979    0.832
```

Regression logistique Ridge

Recherche de la meilleur lambda par cross-validation à 10 plis

```
cv_ridge<- cv.glmnet(x_train, y_train, alpha =0, nfolds =10 , intercept= TRUE, family = "binomial", sta
```

```
plot(cv_ridge)
```




```
cv_ridge$lambda.min
```

```
## [1] 0.009679332
```

- Modèle Ridge optimal

```
model_ridge <- glmnet(x_train, y_train, alpha = 0, lambda=cv_ridge$lambda.min, family = "binomial")
```

- Prédiction

```r
pred_ridge_train=predict(model_ridge,newx = x_train,type = "response")
pred_ridge_train_class=if_else(pred_ridge_train >0.5,"dead","alive")


pred_ridge=predict(model_lasso,newx = x_test,type = "response")

pred_ridge_class=if_else(pred_ridge >0.5,"dead","alive")
```

- Evaluation des performances

```r
evaluation_prediction(data_train_bal2$dead,pred_ridge_train_class)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.6653  0.719     0.657    0.687
```

```r
evaluation_prediction(data_test$dead,pred_ridge_class)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.7188  0.723     0.979    0.832
```

Elasticnet

```r
cv_elastic<- cv.glmnet(x_train, y_train, alpha =0.3, nfolds =10 , intercept= TRUE, family = "binomial",
```

- Modèle elasticnet optimal

```r
model_elastic <- glmnet(x_train, y_train, alpha = 0.3, lambda=cv_elastic$lambda.min, family = "binomial
```

- Prédiction

```r
pred_elastic_train=predict(model_elastic,newx = x_train,type = "response")
pred_elastic_train_class=if_else(pred_elastic_train >0.5,"dead","alive")


pred_elastic=predict(model_elastic,newx = x_test,type = "response")

pred_elastic_class=if_else(pred_elastic >0.5,"dead","alive")
```

- Evaluation des performances

```r
evaluation_prediction(data_train_bal2$dead,pred_elastic_train_class)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.6682  0.719     0.661    0.689
```

```r
evaluation_prediction(data_test$dead,pred_elastic_class)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.7192  0.723     0.979    0.832
```

Autres modèles d'apprentissage automatique pour les problèmes de classification

Machine à vecteur de support (SVM)

```r
library(e1071)

svm_model <- svm(dead ~ .,data=data_train, kernel = "linear", probability=TRUE)
```

- Prédiction

```r
pred_svm_train=predict(svm_model,data_train_bal2[,-1])

pred_svm_test=predict(svm_model,data_test[,-1])
```

- Evaluation des performances

```r
evaluation_prediction(data_train_bal2$dead,pred_svm_train)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.5106      1     0.511    0.676
```

```r
evaluation_prediction(data_test$dead,pred_svm_test)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.9603      1      0.96     0.98
```

K plus proche voisin

```r
library(class)

k_values <- c(1:9)

# Fonction pour évaluer la performance du modèle KNN pour une valeur donnée de k
knn_cv <- function(k) {
  model <- knn.cv(train = x_train,
                  cl = y_train,
                  k = k)
  predictions <- knn(train = x_train,
                     test = x_test,
                     cl = y_train,
                     k = k)
```

```
  accuracy <- sum(predictions == data_test$dead) / nrow(data_test)
  return(accuracy)
}


?knn

# Appliquer la validation croisée pour chaque valeur de k

accuracies <- sapply(k_values, knn_cv)

# Trouver la meilleure valeur de k

best_k <- k_values[which.max(accuracies)]

cat("Meilleur k trouvé:", best_k)
```

```
## Meilleur k trouvé: 1
```

l'hyperparamètre optimal pour le modèle est k=1

```
k_nn_model <- knn(train =x_train,
              test = x_test,
              cl =y_train,
              k = 1,
             prob = TRUE)
```

- Evaluation des performances

```
evaluation_prediction(data_test$dead, k_nn_model)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.8776  0.908     0.962    0.934
```

Modèle ensembliste : Random Forest

- Construction du modèle

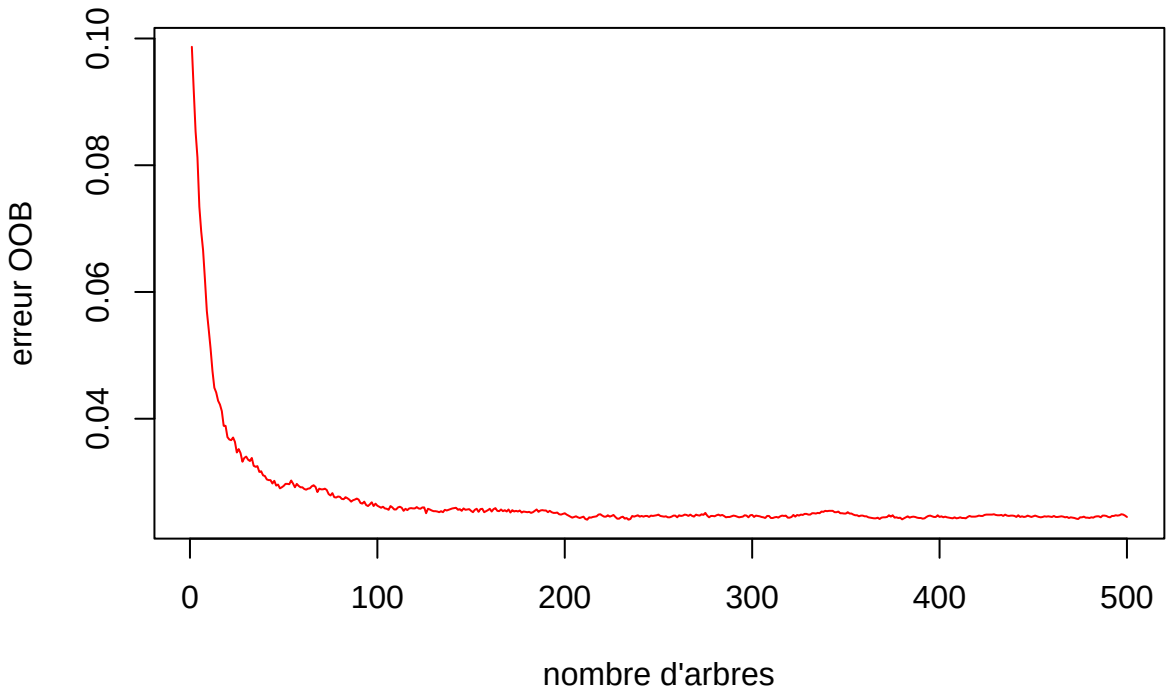
```
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
model_rf=randomForest(x_train,y_train)
```

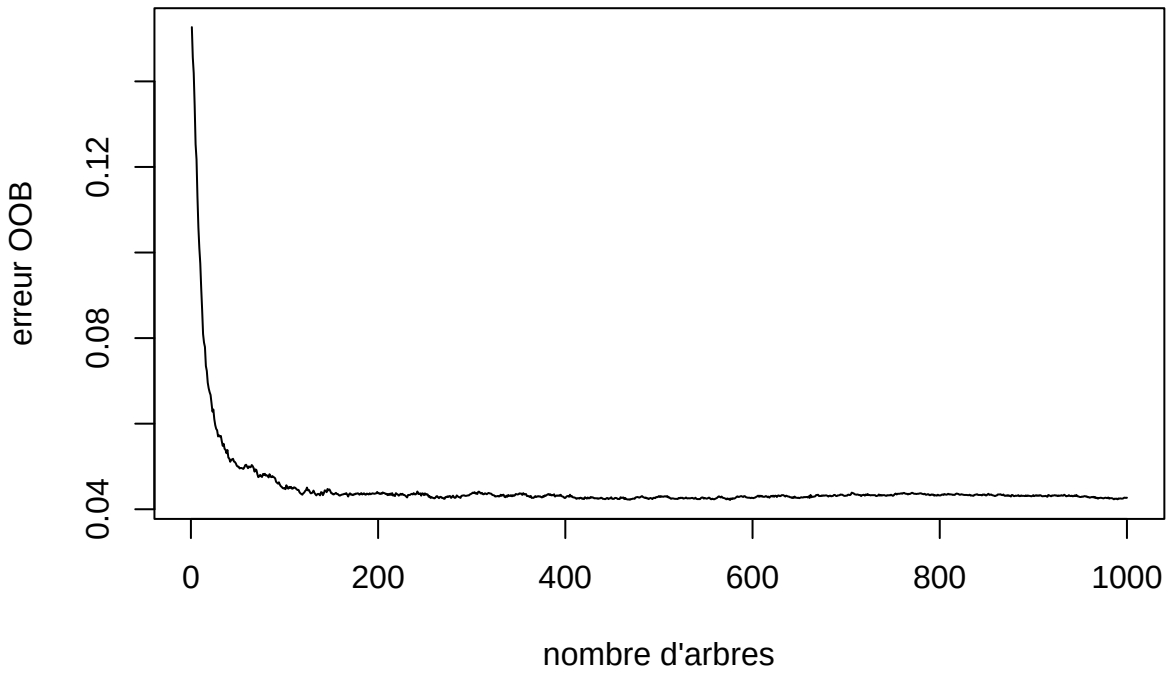
```
plot(model_rf$err.rate[, 1], type = "l", xlab = "nombre d'arbres", ylab = "erreur OOB", col="red")
```




```
set.seed(123)
model_rf_optimal<- randomForest(x_train,y_train, ntree=1000, mtry=4) #mtry nombre de variables testées
print(model_rf_optimal)
```

```
##
## Call:
##  randomForest(x = x_train, y = y_train, ntree = 1000, mtry = 4)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 4.27%
## Confusion matrix:
##        a live dead class.error
## a live   8769  714 0.075292629
## dead       79 9012 0.008689913
```

```r
plot(model_rf_optimal$err.rate[, 1], type = "l", xlab = "nombre d'arbres", ylab = "erreur OOB")
```



- predictions avec le modèle optimal

```r
# sur le data train
predicted_train_rf=predict(model_rf_optimal,newdata = x_train)

# sur le test set

predicted_test_rf=predict(model_rf_optimal,newdata = x_test)
```

- Evaluation des performances

```r
evaluation_prediction(data_train_bal2$dead,predicted_train_rf)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.9659   0.94     0.992    0.966
```

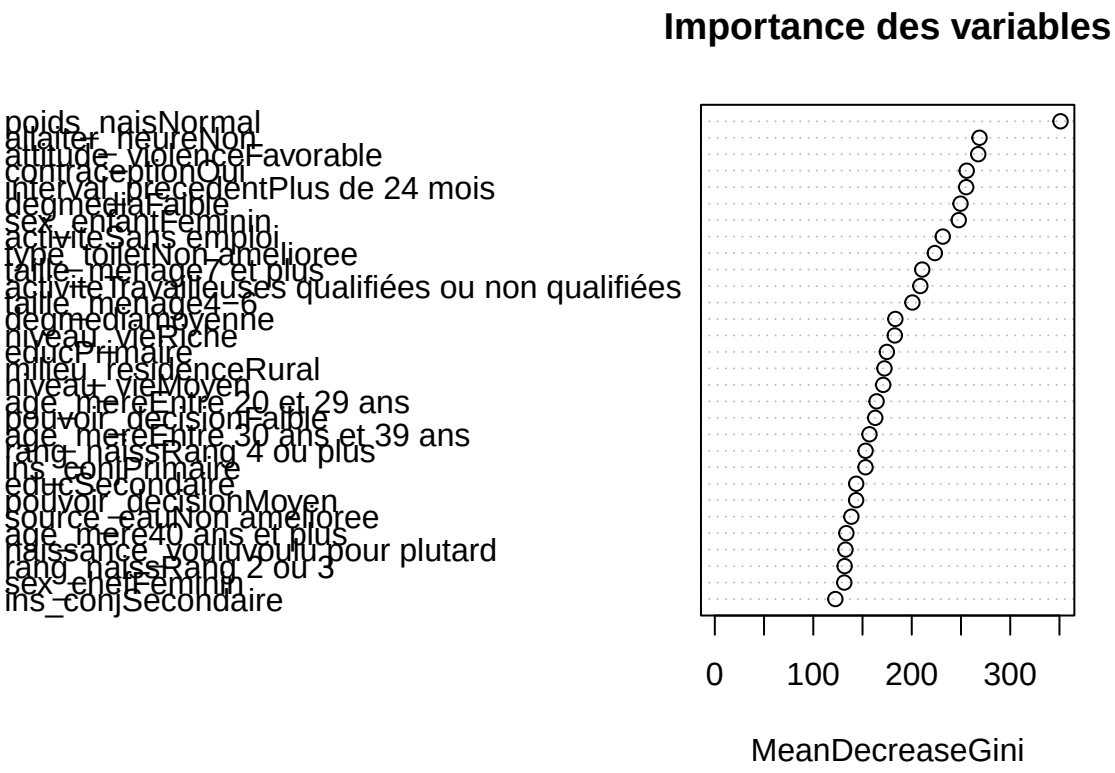
```r
evaluation_prediction(data_test$dead,predicted_test_rf)
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.8947  0.926     0.963    0.944
```

Il existe plusieurs autres indicateurs dans le modèle Random Forest que l'on peut utiliser pour comprendre l'importance de chaque variable dans la prédiction. il s'agit par exemple, la valeur de GINI, la valeur de SHAPLEY etc.

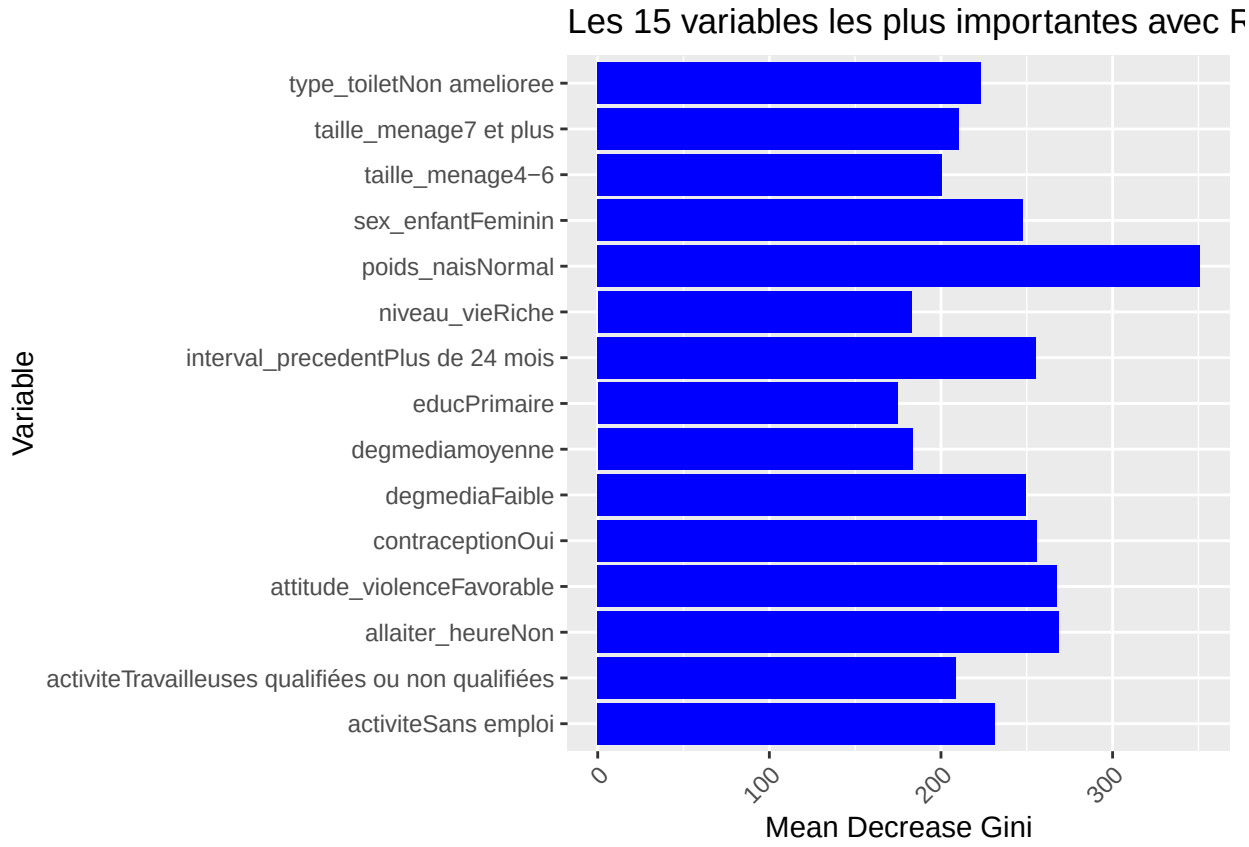
- Visualisation de l'importance des variables

```
varImpPlot(model_rf_optimal, type =2, main = "Importance des variables")
```




```
# Extraire les mesures d'importance
importance <- importance(model_rf_optimal)
importance <- as.data.frame(importance)
importance$variable<- rownames(importance)
```

```
ggplot(importance %>% arrange(desc(MeanDecreaseGini)) %>% head(15)) +
  geom_bar(aes(x = MeanDecreaseGini, y=variable),stat = "identity", fill = "blue") +
  labs(x = "Mean Decrease Gini", y = "Variable") +
  ggtitle("Les 15 variables les plus importantes avec RF") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Resumé des performances pour l'ensemble des modèles ici developés

```
glm=evaluation_prediction(data_test$dead,predicted_class_test)
ridge=evaluation_prediction(data_test$dead,pred_ridge_class)
lasso=evaluation_prediction(data_test$dead,pred_lasso_class)
elasticnet=evaluation_prediction(data_test$dead,pred_elastic_class)
svm=evaluation_prediction(data_test$dead,pred_svm_test)
knn=evaluation_prediction(data_test$dead,k_nn_model)
Randonforest=evaluation_prediction(data_test$dead,predicted_test_rf)


metrics=rbind(glm,ridge,lasso,elasticnet,svm,knn,Randonforest)
metrics
```

```
##   Taux_de_bon_classement Rappel Précision F1_Score
## 1                 0.7180  0.722     0.979    0.831
## 2                 0.7188  0.723     0.979    0.832
## 3                 0.7188  0.723     0.979    0.832
## 4                 0.7192  0.723     0.979    0.832
## 5                 0.9603  1.000     0.960    0.980
## 6                 0.8776  0.908     0.962    0.934
## 7                 0.8947  0.926     0.963    0.944
```

```r
model=c("Glm","Ridge","Lasso","Elasticnet","SVM","knn","Randonforest")

cbind(model,metrics)
```

```
##          model Taux_de_bon_classement Rappel Précision F1_Score
## 1          Glm                 0.7180  0.722     0.979    0.831
## 2        Ridge                 0.7188  0.723     0.979    0.832
## 3        Lasso                 0.7188  0.723     0.979    0.832
## 4   Elasticnet                 0.7192  0.723     0.979    0.832
## 5          SVM                 0.9603  1.000     0.960    0.980
## 6          knn                 0.8776  0.908     0.962    0.934
## 7 Randonforest                 0.8947  0.926     0.963    0.944
```

En conclusion on constate que dans l'ensemble les algorithmes d'apprentissage automatique de type *boite noire* tendent à mieux performer au vue des métriques ici considérées que les modèles de régression. En réalité, le choix définitif du meilleur modèle doit tenir compte non seulement des métriques de performances mais aussi des objectifs de l'étude.