Algorithmes d'apprentissage et modèles statistiques: Un exemple de régression logistique régularisée et de validation croisée pour prédire le décrochage scolaire

Charles-Édouard Giguère Éric Lacourse Véronique Dupéré

2020-06-29

Contents

Abstract: Code des analyses présentées dans la partie 2 du chapitre 23: régression logistique régularisée

Keywords: Apprentissage machine, Régularisation, Validation croisée

2.1. Description de l'échantillon, du devis et des variables

2.1.1. Échantillon

Les données ont été recueillies initialement auprès de n = 6773 d'adolescents provenant de 12 écoles où le taux de décrochage est particulièrement élevé, autour de 36%, afin de mesurer un ensemble de facteurs de risque du décrochage scolaire. Au total, 10 des 12 écoles étaient situées dans des quartiers défavorisés. Un sous-échantillon a par la suite été invité à une entrevue afin d'établir les stresseurs auxquels les adolescents étaient exposés. L'objectif était d'interviewer 45 adolescents par école (pour un total de n = 545), suivant un devis avec cas témoins appariés. D'abord, 15 élèves qui venaient de décrocher de l'école ont été interviewés. Ensuite, 15 élèves appariés ayant un profil initial de risques similaire, mais qui persévéraient ont également été interviewés. Finalement, 15 autres élèves « normatifs », également persévérants, qui avaient un niveau moyen de risque ont été interviewés.

2.1.2. Variables

La variable dépendante est une variable dichotomique (codée 0 = non/1 = oui) représentant le fait qu'un élève a décroché de l'école ou non. Un élève est considéré comme décrocheur s'il remplit au moins une des trois conditions suivantes : 1) avoir avisé officiellement de la cessation de ses études avant d'obtenir son diplôme d'études secondaires ou DES, 2) avoir été transféré au système d'éducation aux adultes, 3) être

absent pendant plus d'un mois de l'école sans avoir avisé la direction des motivations sous-jacentes. Pour plus de détails sur les variables et le devis, voir l'article original de Dupéré et al. (2018)Une particularité de la structure des données est que la grande proportion des variables sont ordinales et recodées en variables factices (dummy) dichotomiques. En général, les effets de la régulation sont plus marqués : 1) avec des variables intervalles/ratio puisqu'elles ont une plus grande variance et 2) en présence de multicollinéarité. Une autre caractéristique des données est que nous ne sommes pas en contexte de haute dimensionnalité puisque nous avons 25 variables pour 1000 participants, donc p « n. En dernier lieu, l'étude originale est de nature confirmatoire (hypothético-déductive) et non exploratoire (inductive), ce qui favorisera les algorithmes les moins flexibles, comme la régression logistique « classique ».

Description des variables indépendantes introduites dans le modèle de régression logistique régularisée provenant de l'étude de Dupéré et al. (2018)

Nom des variables - Types de variables - Nom des variables dans le fichier de données

1. Sexe - Dichotomique - MALE
2. Âge - Intervalle - AGE
3. Parent immigré - Dichotomique - PAR_IMM
4. Ethnicité - Dichotomique - MINORITY
5. Niveau de scolarité parents - Intervalle - SCOLMAX
6. Mère en emploi - Dichotomique - TRAVAILM
7. Père en emploi - Dichotomique - TRAVAILP
8. Parents séparés - Dichotomique - PAR_SEP
9. Adaptation scolaire - Intervalle - ADAPT
10. Risque décrochage scolaire - Intervalle - SRDQ
11. Difficultés chroniques sévères - Intervalle - CHRONSEVACT
12. Stresseurs sévères 0-3 mois - Dichotomique - SEVER03DICO
13. Stresseurs sévères 3-6 mois - Dichotomique - SEVER36DICO
14. Stresseurs sévères 6-9 mois - Dichotomique - SEVER69DICO
15. Stresseurs sévères 9-12 mois - Dichotomique - SEVER912DICO
16. Stresseurs modérés 0-3 mois - Dichotomique - MODER203DICO
17. Stresseurs modérés 3-6 mois - Dichotomique - MODER236DICO
18. Stresseurs modérés 6-9 mois - Dichotomique - MODER269DICO
19. Stresseurs modérés 9-12 mois - Dichotomique - MODER2912DICO
20. Stresseurs faibles 0-3 mois - Dichotomique - LOW203DICO
21. Stresseurs faibles 3-6 mois - Dichotomique - LOW236DICO
22. Stresseurs faibles 6-9 mois - Dichotomique - LOW269DICO
23. Stresseurs faibles 9-12 mois - Dichotomique - LOW2912DICO
24. Stresseurs distaux sévères - Intervalle/ratio - EVDISTSEV
25. Stresseurs distaux modérés - Intervalle/ratio - EVDISTMOD

2.2. Objectifs de l'analyse

L'objectif principal de cette analyse est de sélectionner un modèle de régression logistique, de manière exploratoire/inductive, en utilisant des techniques qui sont particulières à l'apprentissage automatique afin de potentiellement prédire le décrochage scolaire avec la plus grande justesse prédictive possible à partir des 25 variables indépendantes. Nous utilisons des données simulées à partir de l'échantillon initial.En résumé, la tâche de classification consiste à trouver à la fois le nombre optimal de prédicteurs du décrochagescolaire et l'algorithme de régularisation qui permettra le meilleur ajustement du modèle aux données, compte tenu de la spécificité des variables introduites dans le modèle.

2.3. et 2.4 Régression logistique régularisée : procédures et modèles + interprétations et taleaux de résultats

La suite du document montre le code utilisé (originalement dans le logiciel RStudio) pour la sélection du modèle conformément aux procédures décrites dans la partie 2.3 et aux résultats montrés dans la partie 2.4 du chapitre

```
#Téléchargement des packages nécessaire à l'analyse (si vous installez ces packages pour la première fo

#install.packages("CUFF") #Package CUFF (Charles's Utility Function using Formula) pour affichage des v
#install.packages ("dplyr") #Package dplyr pour manipulation flexible des données
#install.packages("ggplot2") #Package ggplot2 pour création de graphiques
#install.packages("haven") #Package haven pour importer des données d'autres formats dans R
#install.packages("knitr") #Package knitr pour production de tableau
#install.packages("xtable") #Package xtable pour production de tableau
#install.packages("pairwise") #Package xtable pour production de tableau


require(dplyr, quietly = TRUE, warn.conflicts = FALSE)
require(ggplot2, quietly = TRUE, warn.conflicts = FALSE)
require(CUFF, quietly = TRUE, warn.conflicts = FALSE)
require(haven, quietly = TRUE, warn.conflicts = FALSE)
require(knitr, quietly = TRUE, warn.conflicts = FALSE)
require(xtable, quietly = TRUE, warn.conflicts = FALSE)
require(pairwise, quietly = TRUE, warn.conflicts = FALSE)

opts_chunk$set(echo = TRUE, prompt = TRUE, comment = "", cache = TRUE)
options(xtable.comment = FALSE)

#install.packages("glmnet", quietly = TRUE, warn.conflicts = FALSE, dependencies = TRUE)
#install.packages("latex2exp", quietly = TRUE, warn.conflicts = FALSE, dependencies = TRUE)

require(glmnet)
```

```
## Loading required package: glmnet

## Loading required package: Matrix

## Loaded glmnet 4.0-2
```

```
require(latex2exp)
```

```
## Loading required package: latex2exp
```

```
> #Télécharger le fichier de données à partir de github
> SD.csv <- "https://github.com/ericlacourse/Code-chapitre-23.git"
>
> #Lire le fichier de données téléchargé depuis github (vérification du téléchargement des données et d
> library(readr)
> SD.df <- read.csv("SD.csv")
> ls(SD.df)
```

```
 [1] "ADAPT"          "AGE"            "AIMES"          "AMB"
 [5] "CHRONSEVACT"    "DOUBLE"         "ECOLE"          "EVDISTMOD"
 [9] "EVDISTSEV"      "ID"             "IMP"            "LOW203DICO"
[13] "LOW236DICO"     "LOW269DICO"     "LOW2912DICO"    "MALE"
[17] "MINORITY"       "MODER203DICO"   "MODER236DICO"   "MODER269DICO"
[21] "MODER2912DICO"  "NOTES_FR"       "NOTES_MATH"     "PAR_IMM"
[25] "PAR_SEP"        "PR_AUTRES"      "SCOLMAX"        "SEVER03DICO"
[29] "SEVER36DICO"    "SEVER69DICO"    "SEVER912DICO"   "SRDQ"
[33] "STATUT"         "TRAVAILM"       "TRAVAILP"       "X"
```

Développement des modèles à partir de l'échantillon d'entrainement On doit idéalement éviter le sur- ou le sous-apprentissage, c'est-à-dire d'avoir une excellente classification au sein de l'échantillon d'entraînement, mais une mauvaise classification avec de nouvelles données . Une solution pour éviter ce problème est de diviser aléatoirement l'échantillon en deux parties, 70% de données d'entraînement et 30% de données « test » (soit des nouvelles données). Cette proportion est un peu arbitraire, mais l'idée est de garder un échantillon d'entraînement le plus grand possible pour pouvoir développer un modèle tout en ayant un échantillon « test » assez grand pour valider le modèle. Ainsi, l'échantillon est séparé en deux sous-échantillons: 1. Fichier d'entraînement (TRAIN.df ; N = 700) 2. Fichier de test (TEST.df ; N = 300)

```
> #Diviser l'ensemble de données pour créer un sous-ensemble de données d'entrainement et un sous-ensem
> set.seed(1234)
> ECH.TRAIN <- sample(1:1000, 700)
> TRAIN.df <- SD.df[ECH.TRAIN,]
> TEST.df <- SD.df[-ECH.TRAIN,]
```

```
> #Standardisation des variables
> TRAIN.df[,-(2:4)]  <- scale(TRAIN.df[,-(2:4)])
> TEST.df[,-(2:4)]  <- scale(TEST.df[,-(2:4)])
>
> TRAIN.df <- as.data.frame(TRAIN.df)
> TEST.df <- as.data.frame(TEST.df)
```

```
> #Lire l'ensemble de données d'entrainement pour vérifier la réussite de l'étape précédente
> ls (TRAIN.df)
```

```
 [1] "ADAPT"          "AGE"            "AIMES"          "AMB"
 [5] "CHRONSEVACT"    "DOUBLE"         "ECOLE"          "EVDISTMOD"
 [9] "EVDISTSEV"      "ID"             "IMP"            "LOW203DICO"
[13] "LOW236DICO"     "LOW269DICO"     "LOW2912DICO"    "MALE"
[17] "MINORITY"       "MODER203DICO"   "MODER236DICO"   "MODER269DICO"
[21] "MODER2912DICO"  "NOTES_FR"       "NOTES_MATH"     "PAR_IMM"
[25] "PAR_SEP"        "PR_AUTRES"      "SCOLMAX"        "SEVER03DICO"
[29] "SEVER36DICO"    "SEVER69DICO"    "SEVER912DICO"   "SRDQ"
[33] "STATUT"         "TRAVAILM"       "TRAVAILP"       "X"
```

```
> #Lire l'ensemble de données de test pour vérifier la réussite de l'étape précédente
> ls (TEST.df)
```

```
 [1] "ADAPT"          "AGE"            "AIMES"          "AMB"
 [5] "CHRONSEVACT"    "DOUBLE"         "ECOLE"          "EVDISTMOD"
 [9] "EVDISTSEV"      "ID"             "IMP"            "LOW203DICO"
[13] "LOW236DICO"     "LOW269DICO"     "LOW2912DICO"    "MALE"
```

```
[17] "MINORITY"      "MODER203DICO"  "MODER236DICO"  "MODER269DICO"
[21] "MODER2912DICO" "NOTES_FR"      "NOTES_MATH"    "PAR_IMM"
[25] "PAR_SEP"       "PR_AUTRES"     "SCOLMAX"       "SEVER03DICO"
[29] "SEVER36DICO"   "SEVER69DICO"   "SEVER912DICO"  "SRDQ"
[33] "STATUT"        "TRAVAILM"      "TRAVAILP"      "X"
```

Par la suite, l'échantillon d'entraînement est aussi divisé aléatoirement en dix sous-échantillons de 70 unités pour permettre la validation croisée à k plis ainsi que la sélection du paramètre alpha ou lambda afin de permettre une régularisation adéquate. Les partitions et la fonction crossval sont construites afin de préparer la validation croisée.

```
> #Division des données d'entrainement en 10 groupes de 70 individus (observations)
> PARTITION = sample(rep(1:10, rep(70,10)),700)
>
> #Création de la fonction crossval pour la validation croisée à 10-plis
> crossval <- function(mod){
+   f1 <- function(x){
+     modi = update(mod, data = TRAIN.df[!(PARTITION %in% x),])
+     table(1*(predict(modi, newdata = TRAIN.df[PARTITION %in% x,],
+                      type = "resp")>0.5),
+           TRAIN.df[(PARTITION %in% x),"STATUT"])
+   }
+   CVT <- mapply(f1, x = 1:10)
+   as.table(matrix(apply(CVT, 1, sum), 2, 2,
+                   dimnames = list(c("P.ND","P.D"),
+                                   c("T.ND","T.D"))))
+ }
```

Une manière adéquate de commencer les analyses est d'estimer un modèle de régression logistique « classique » en utilisant les 25 prédicteurs. En premier lieu, les données de notre échantillon seront donc modélisées à partir d'une régression logistique « classique ». Les 25 variables standardisées, potentiellement associées au décrochage, sont introduites dans le modèle.

```
> #Un modèle additif (sans régularisation) de régression logistique est ajusté sur l'échantillon d'entr
> #Régression logistique classique; données d'entraînement
> var.model <- c("MALE", "AGE", "PAR_IMM", "MINORITY", "SCOLMAX", "TRAVAILM", "TRAVAILP", "PAR_SEP", "A
+                "SRDQ", "EVDISTSEV", "EVDISTMOD","SEVER03DICO",
+                "SEVER36DICO", "SEVER69DICO", "SEVER912DICO",
+                "MODER203DICO", "MODER236DICO", "MODER269DICO",
+                "MODER2912DICO", "LOW203DICO", "LOW236DICO",
+                "LOW269DICO", "LOW2912DICO", "CHRONSEVACT")
> glm1 <- glmnet(x = TRAIN.df[, var.model] %>% as.matrix, y = TRAIN.df[,"STATUT"], lambda=0, family = "
```

```
> #Visualisation des résultats
> print(glm1)
```

```
Call:  glmnet(x = TRAIN.df[, var.model] %>% as.matrix, y = TRAIN.df[,      "STATUT"], family = "binomial

  Df  %Dev Lambda
1 25 16.07      0
```

```
> predict(glm1, type="coef", "lambda.min", allCoef = TRUE)
```

```
26 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)    0.01223880
MALE           0.11619607
AGE            0.37832290
PAR_IMM       -0.02354991
MINORITY       0.01117619
SCOLMAX        0.28359589
TRAVAILM       0.11628193
TRAVAILP      -0.10514515
PAR_SEP        0.54567272
ADAPT         -0.12379524
SRDQ          -0.04120007
EVDISTSEV      0.26398912
EVDISTMOD     -0.49879199
SEVER03DICO    0.43484789
SEVER36DICO   -0.06016151
SEVER69DICO   -0.45572581
SEVER912DICO  -0.01289725
MODER203DICO  -0.36817345
MODER236DICO   0.36277321
MODER269DICO   0.44343925
MODER2912DICO  0.09907921
LOW203DICO     0.25546687
LOW236DICO     0.12153723
LOW269DICO    -0.23443896
LOW2912DICO   -0.17378213
CHRONSEVACT    0.60983017
```

```
> #Prédiction à l'aide de la régression logistique classique
> glm1p <- predict(glm1, newx = TRAIN.df[,var.model] %>%
+                    as.matrix, s = "lambda.min")
```

```
> #Table de classification montrant la performance prédictive du modèle (fréquences, puis proportions)
> cv0 <- table(1*(glm1p>0), TRAIN.df$STATUT)
> cv0
```

```
   0   1
0 245 105
1 104 246
```

```
> prop.table(cv0)*100
```

```
          0        1
0 35.00000 15.00000
1 14.85714 35.14286
```

```
> sprintf("%.1f%% de bonne classification", sum(diag(prop.table(cv0)))*100)
```

```
[1] "70.1% de bonne classification"
```

Nous pouvons constater que 70,1% des participants sont bien classés en utilisant l'échantillon d'entraînement et la validation croisée à 10-plis pour la régression linéaire classique.

Régression régularisée

Passons maintenant aux modèles avec régularisation.

La régularisation ou estimateurs par rétrécissement consiste à pénaliser la fonction objective (moindre carré) servant à estimer les coefficients. Essentiellement, il s'agit d'estimer les coefficients en donnant une pénalité de façon à réduire la dimension de la régression. Une régression OLS aura p paramètres à estimer. Une régression avec régularisation aura un nombre de degrés de liberté inférieur à p permettant d'avoir un modèle linéairement plus parcimonieux. Pour estimer les coefficients d'un modèle régularisé, on utilise la fonction objective suivante:

$$RSS_{shrinkage} = (Y - BX) + \lambda f(B)$$

Nous utilisons trois méthodes correspondant à trois pénalités :

1.La méthode de régularisation de Ridge utilise une pénalité quadratique.

$$RSS_{shrinkage} = (Y - BX) + \lambda \sum_{i=1}^{p} \beta_i^2$$

2.La méthode du Lasso utilise une pénalité en valeur absolue. Cette pénalité fait en sorte que si un coefficient est à 0 pour un λ donné il restera fixé à 0 pour tous les $\lambda^* > \lambda$.

$$RSS_{shrinkage} = (Y - BX) + \lambda \sum_{i=1}^{p} |\beta_i|$$

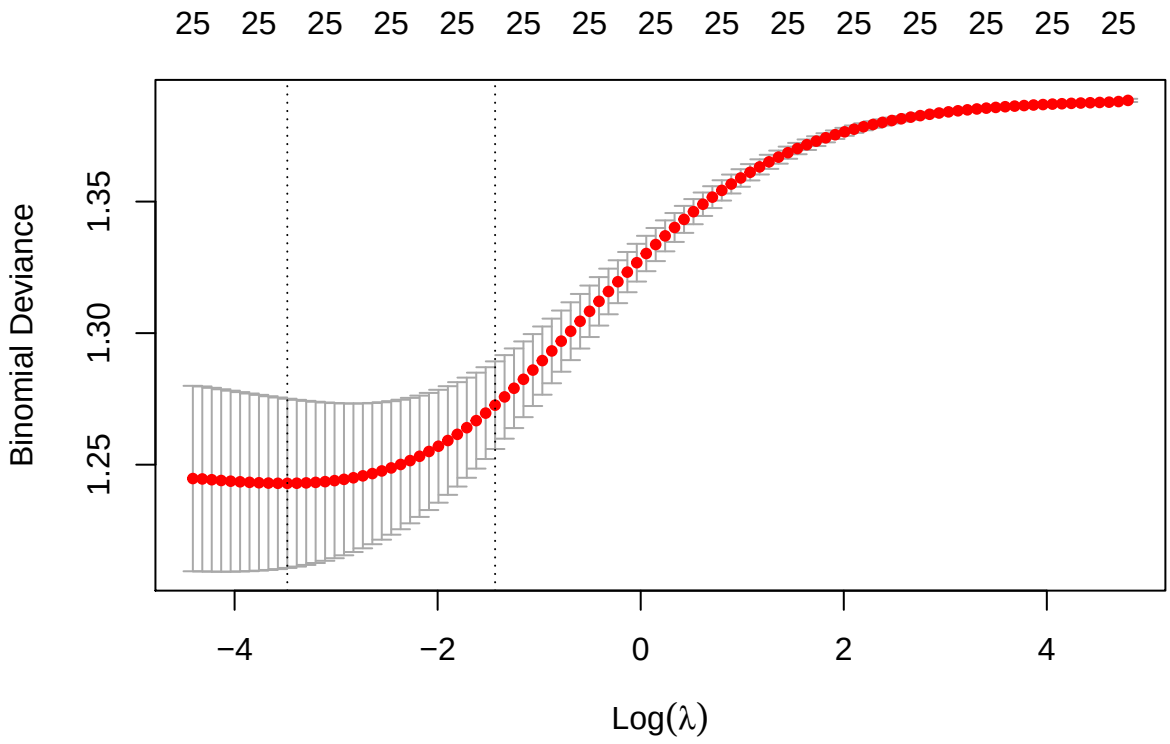
3.La méthode de régularisation elastic net utilise un mélange de deux pénalités. Cette méthode introduit un nouveau paramètre (α) à estimer dans le modèle. En utilisant la paramétrisation suivante, on obtient que $\alpha = 0$ correspond à une régression de Ridge et $\alpha = 1$ correspond à une régression de lasso. Un alpha entre 0 et 1 produit un mélange des deux pénalités.

$$RSS_{shrinkage} = (Y - BX) + +(1 - \alpha)(\sum_{i=1}^{p} \beta_i^2) + (\alpha)(\lambda \sum_{i=1}^{p} |\beta_i|)$$

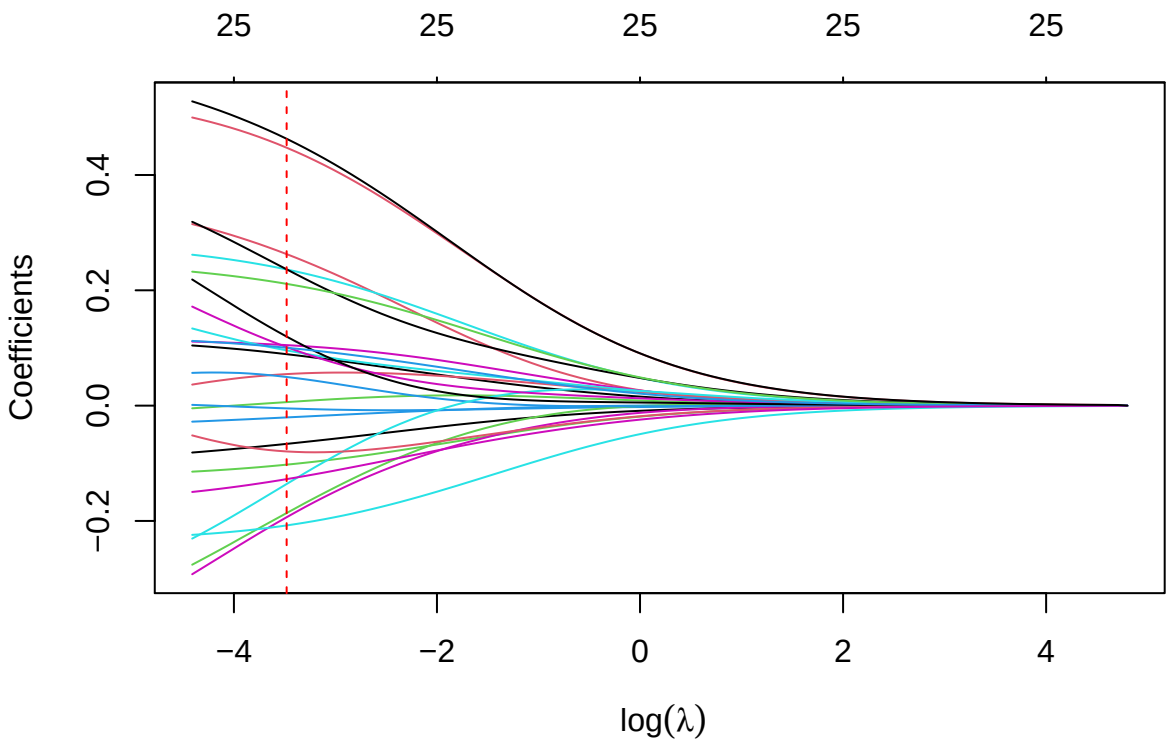
Commençons par essayer la régression logistique avec la régularisation ridge.

```
> ##Régression logistique avec régularisation ridge
> #Sélection du lambda par validation croisée à 10 plis
> var.model <- c("MALE", "AGE", "PAR_IMM", "MINORITY", "SCOLMAX", "TRAVAILM", "TRAVAILP", "PAR_SEP", "AI
+                  "SRDQ", "EVDISTSEV", "EVDISTMOD","SEVER03DICO",
+                  "SEVER36DICO", "SEVER69DICO", "SEVER912DICO",
+                  "MODER203DICO", "MODER236DICO", "MODER269DICO",
+                  "MODER2912DICO", "LOW203DICO", "LOW236DICO",
+                  "LOW269DICO", "LOW2912DICO", "CHRONSEVACT")
```

```
>
> cv.glmn1 <- cv.glmnet(x= TRAIN.df[,var.model] %>% as.matrix,
+                       y = TRAIN.df[,"STATUT"], alpha = 0, nfolds = 10,
+                       foldid = PARTITION, intercept= TRUE,
+                       family = "binomial", standardize = TRUE)
>
> #Visualisation des résultats de la validation croisée avec régularisation ridge (valeur de lambda opt
> plot(cv.glmn1)
```




```
> ##Régression logistique avec régularisation ridge
> glmn1.0 <- glmnet(x = TRAIN.df[, var.model] %>% as.matrix,
+                   y = TRAIN.df[,"STATUT"], alpha = 0, family = "binomial")
>
> #Visualisation: évolution des coefficients selon valeur de lambda avec régularisation ridge + ligne r
> plot(glmn1.0, xvar = "lambda", label = FALSE, xlab = ~ log(lambda))
> abline( v = log(cv.glmn1$lambda.min), col = "red", lty = 2)
```



La validation croisé a été appliquée pour trouver le paramètre lambda. On fait une prédiction basée sur ce modèle, mais la généralisation de la prédiction selon ce modèle sera confirmée à l'aide de l'échantillon de test.

```
> #Prédiction à l'aide de la régression logistique avec régularisation ridge
> glmn1p <- predict(cv.glmn1, newx = TRAIN.df[,var.model] %>%
+                     as.matrix, s = "lambda.min")
```

```
> #Table de classification montrant la performance prédictive du modèle
> cv2 <- table(1*(glmn1p>0), TRAIN.df$STATUT)
> cv2
```

```
    0   1
0 243 100
1 106 251
```

```
> prop.table(cv2)*100
```

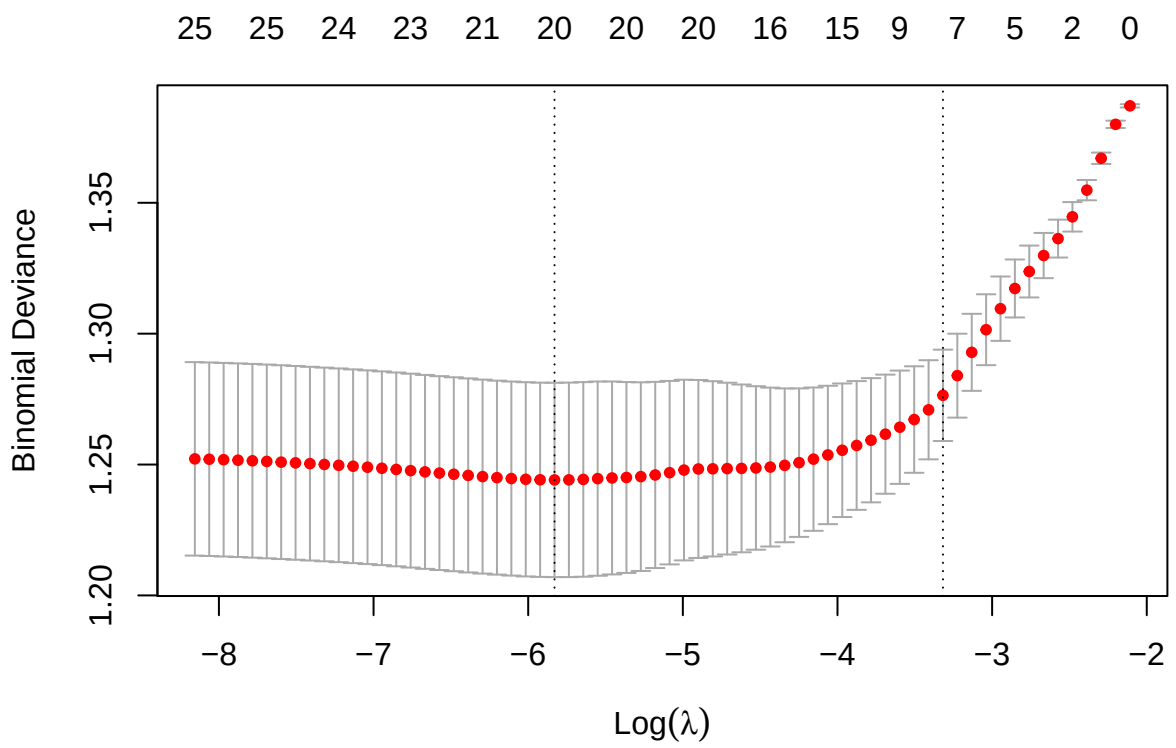
```
          0        1
0 34.71429 14.28571
1 15.14286 35.85714
```

```
> sprintf("%.1f%% de bonne classification", sum(diag(prop.table(cv2)))*100)
```

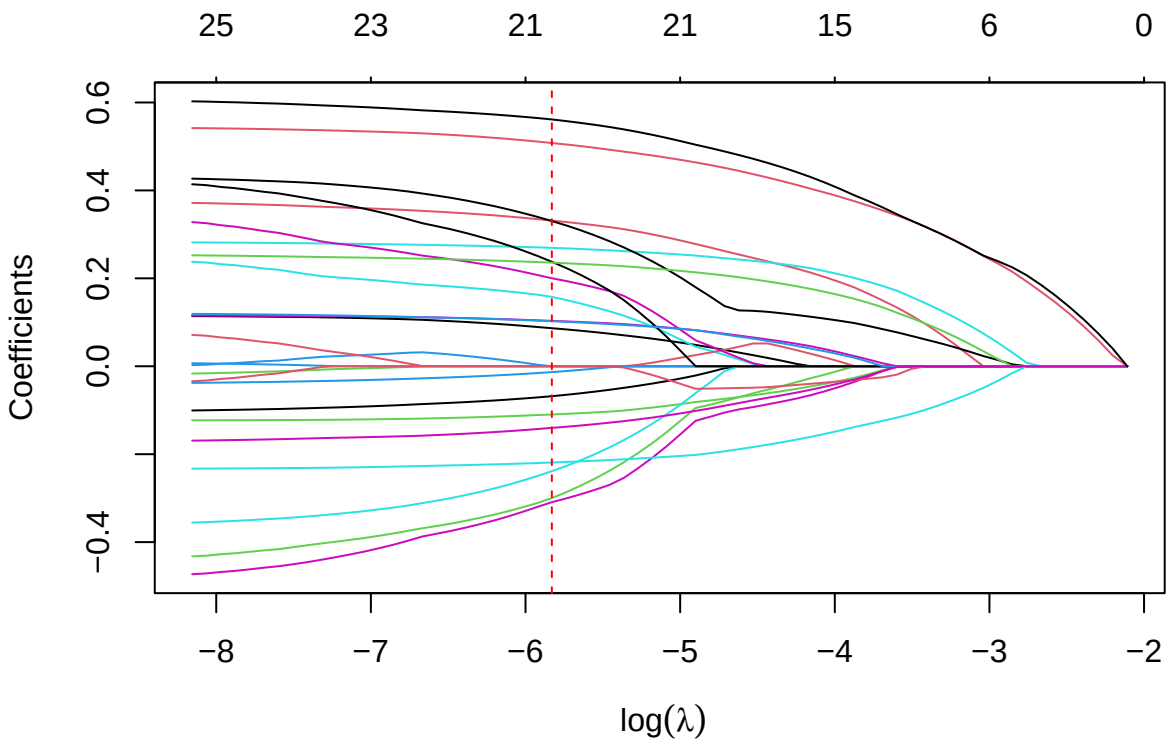
```
[1] "70.6% de bonne classification"
```

On passe maintenant à la régularisation lasso.

```
> ##Régression logistique avec régularisation ridge
> #Sélection du lambda par validation croisée à 10 plis
> cv.glmn2 <- cv.glmnet(x = TRAIN.df[,var.model] %>% as.matrix,
+                       y = TRAIN.df[,"STATUT"], alpha = 1, nfolds = 10,
+                       foldid = PARTITION, family = "binomial")
>
> glmn2 <- glmnet(x = TRAIN.df[,var.model] %>%
+                   as.matrix, y = TRAIN.df[,"STATUT"], alpha = 1, family = "binomial",
+                   lambda = cv.glmn2$lambda.min)
>
> #Visualisation des résultats de la validation croisée avec régularisation lasso (valeur de lambda opt
> plot(cv.glmn2)
```




```
> ##Régression logistique avec régularisation ridge
> glmn2.0 <- glmnet(x = TRAIN.df[,var.model] %>% as.matrix,
+                     y = TRAIN.df[,"STATUT"], alpha = 1, family = "binomial")
>
> #Visualisation: évolution des coefficients selon valeur de lambda avec régularisation lasso + ligne r
> plot(glmn2.0, xvar = "lambda", label = FALSE, xlab = ~log(lambda))
> abline(v = log(cv.glmn2$lambda.min), lty = 2, col = "red")
```



La validation croisé a été appliquée pour trouver le paramètre λ optimal. On fait une prédiction basée sur ce modèle et on compare au vrai statut de décrochage dans les données d'entrainement.

```
> #Prédiction à l'aide de la régression logistique avec régularisation ridge
> glmn2p <- predict(cv.glmn2, newx = TRAIN.df[,var.model] %>%
+                     as.matrix, s = "lambda.min")
```

```
> #Table de classification montrant la performance prédictive du modèle
> cv3 <- table(1*(glmn2p>0), TRAIN.df$STATUT)
> cv3
```

```
     0   1
  0 242 104
  1 107 247
```

```
> prop.table(cv3)*100
```

```
          0        1
  0 34.57143 14.85714
  1 15.28571 35.28571
```

```
> sprintf("%.1f%% de bonne classification", sum(diag(prop.table(cv3)))*100)
```

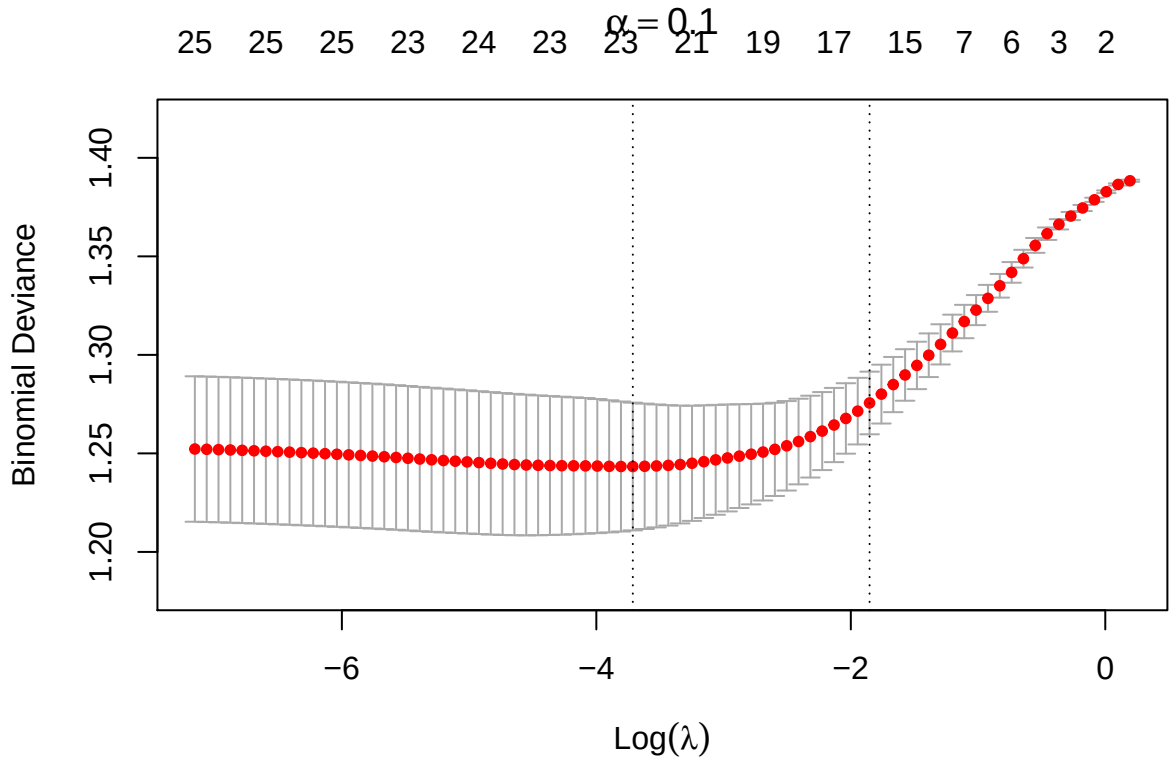
```
[1] "69.9% de bonne classification"
```

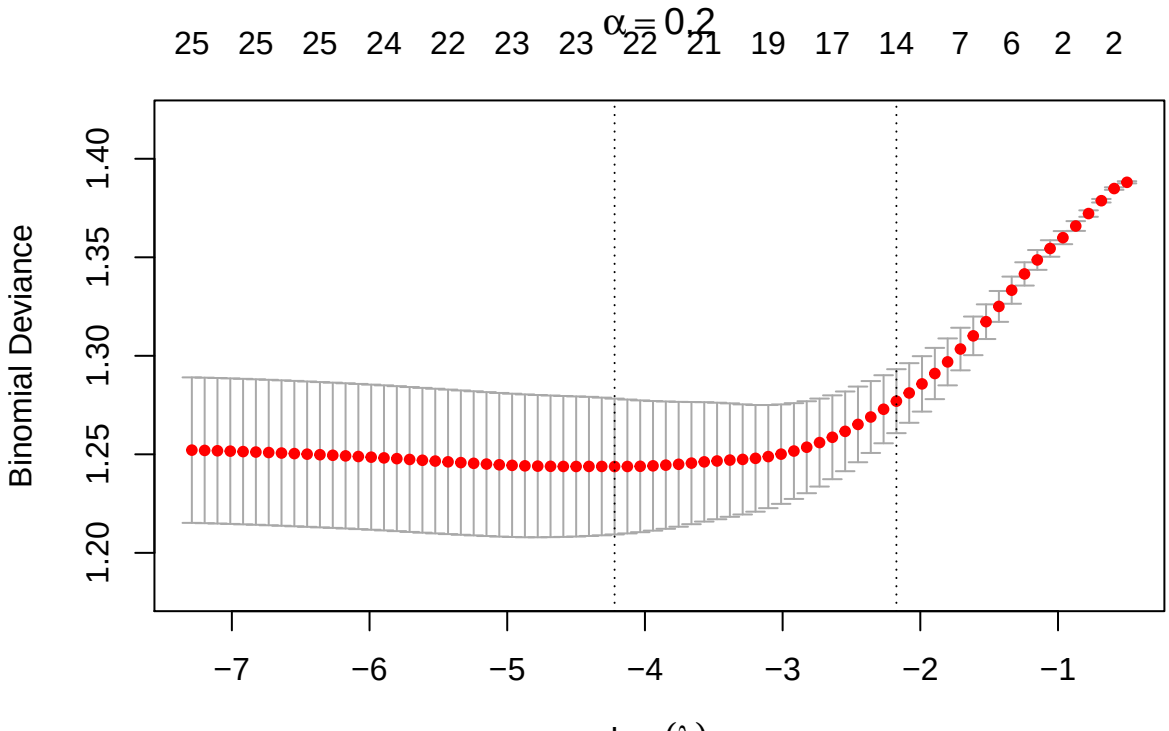
On complète finalement avec la régularisation elastic-net, qui combine les deux méthodes précédentes. Pour cette dernière section, on veut calculer un compromis entre le modèle lasso et le modèle ridge. Il faut donc estimer un paramètre supplémentaire ($\alpha \in (0, 1)$).

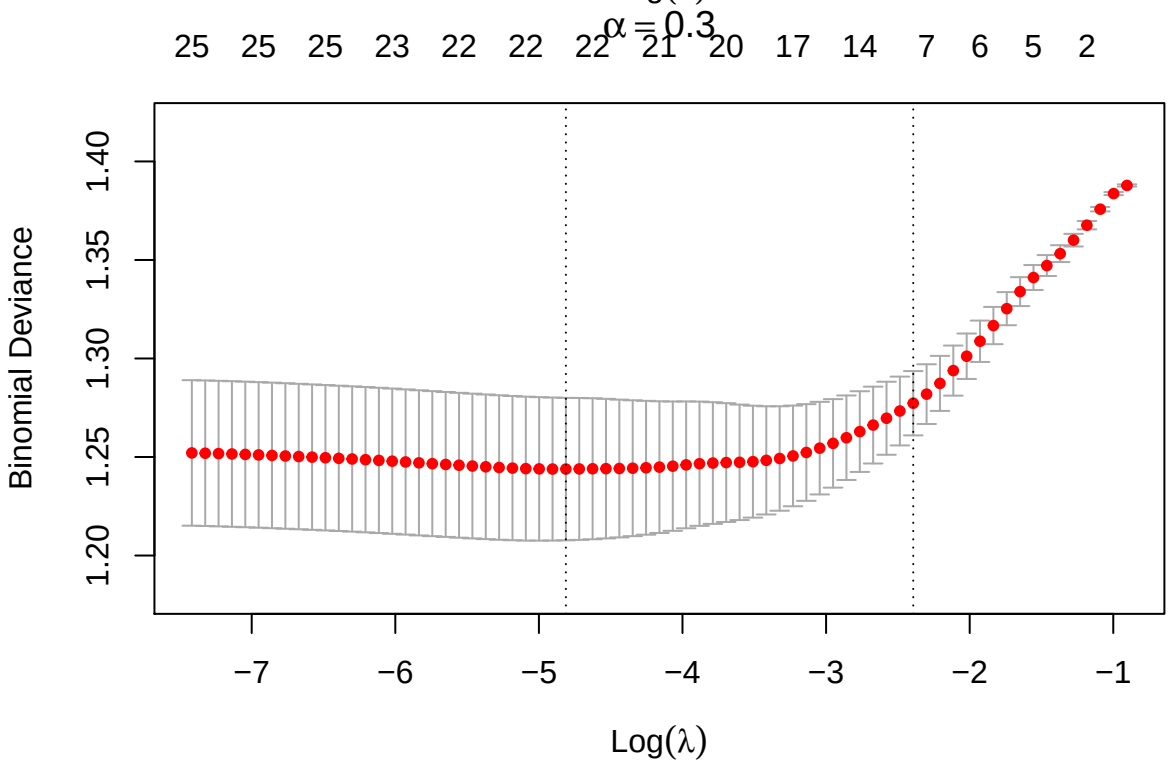
```
> #Ajustement de la matrice
> layout(matrix(1:10,3,3, byrow = TRUE))
```

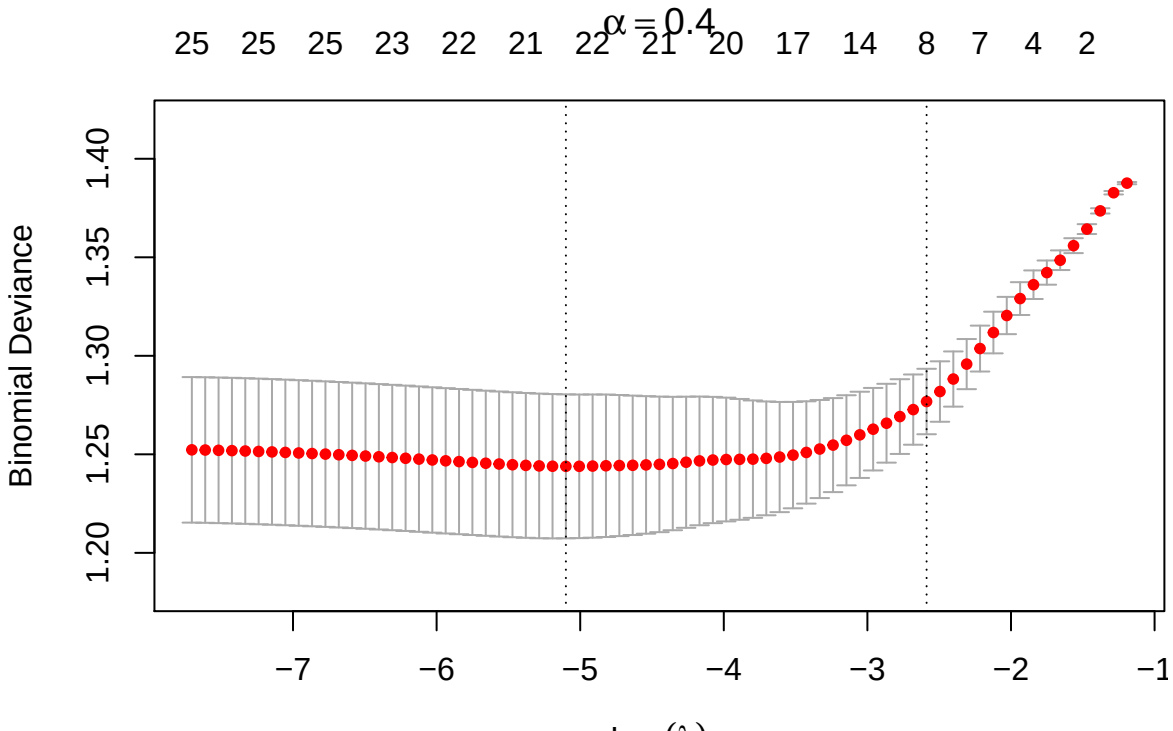
```
Warning in matrix(1:10, 3, 3, byrow = TRUE): data length [10] is not a sub-
multiple or multiple of the number of rows [3]
```

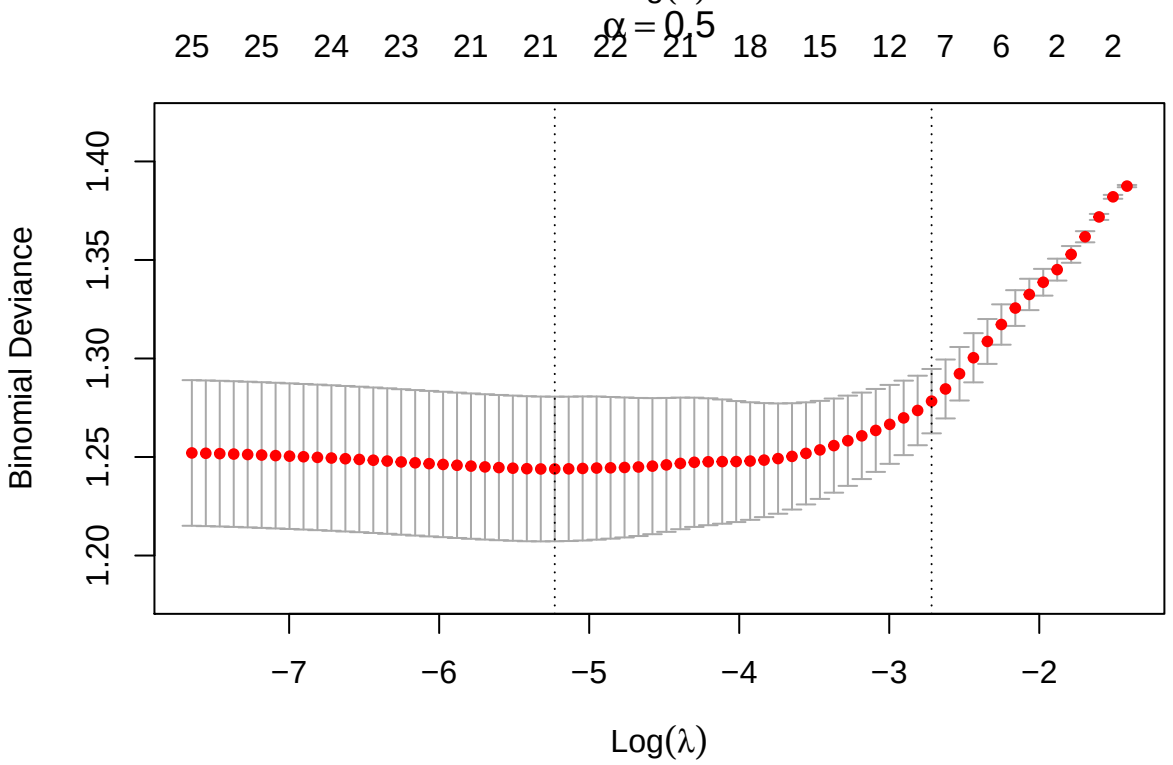
```
> #Validation croisée à 10-plis pour obtention de la valeur optimale de lambda selon la valeur d'alpha
> cv.glmn3 <- list()
>
> for(al in seq(0.1,0.9,0.1)){
+     cv.glmn3[[sprintf("%.1f",al)]] <-
+     cv.glmnet(x = TRAIN.df[,var.model] %>% as.matrix,
+     y = TRAIN.df[,"STATUT"], nfolds = 10, foldid = PARTITION,
+     alpha = al, family = "binomial")
+ plot(cv.glmn3[[sprintf("%.1f",al)]],
+ main = latex2exp::TeX(sprintf("$\\alpha = %.1f$",al)), ylim = c(1.18, 1.42))
+ }
```

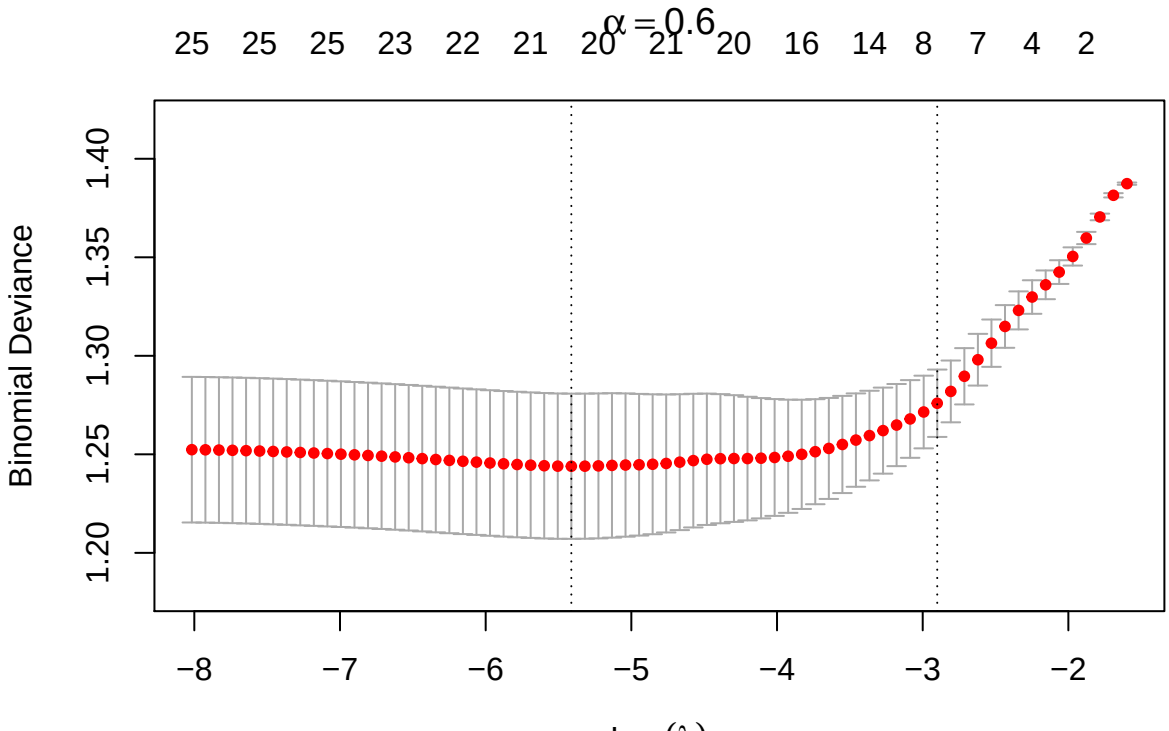


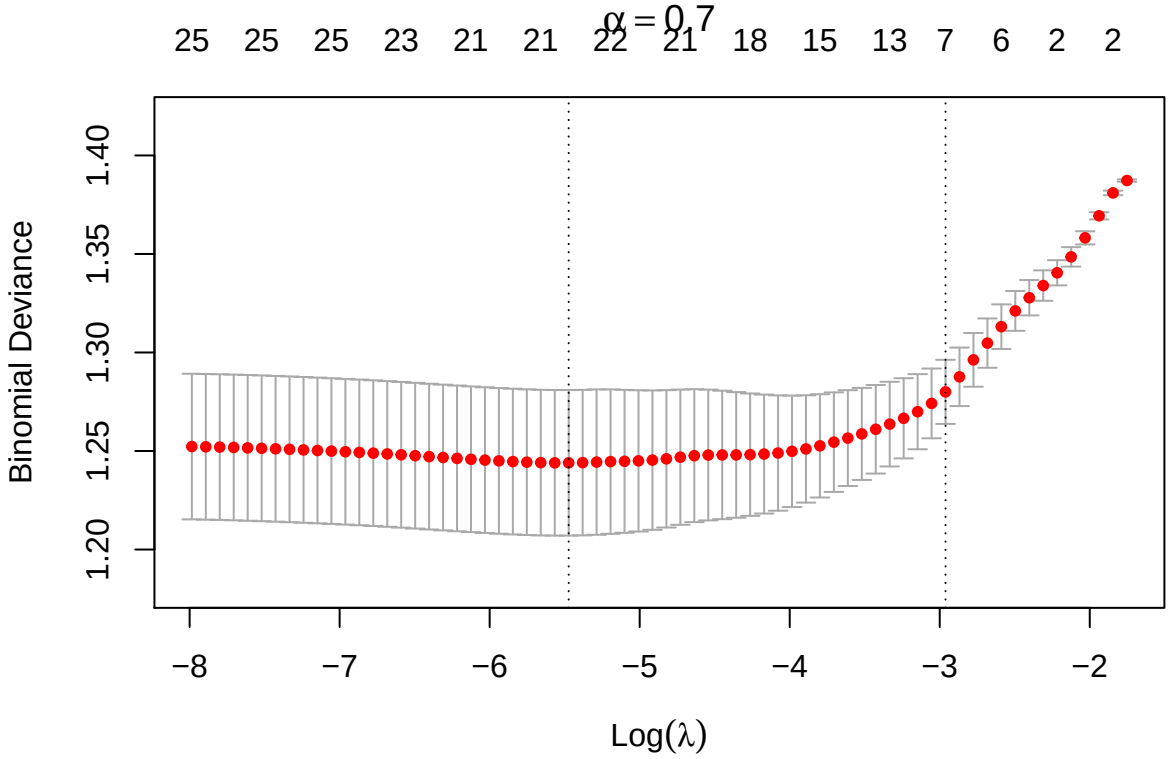

α = 0.2
25 25 25 24 22 23 23 22 21 19 17 14 7 6 2 2
Binomial Deviance
1.20
1.25
1.30
1.35
1.40
−7 −6 −5 −4 −3 −2 −1
Log(λ)

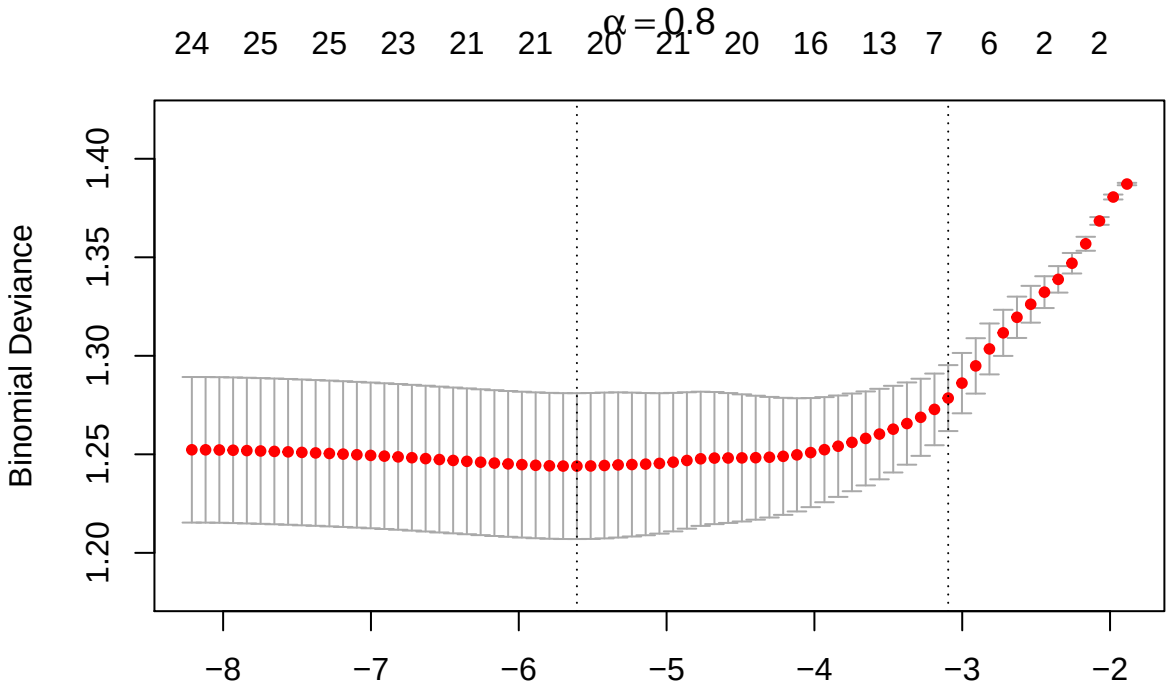

α = 0.3
25 25 25 23 22 22 22 21 20 17 14 7 6 5 2
Binomial Deviance
1.20
1.25
1.30
1.35
1.40
−7 −6 −5 −4 −3 −2 −1
Log(λ)

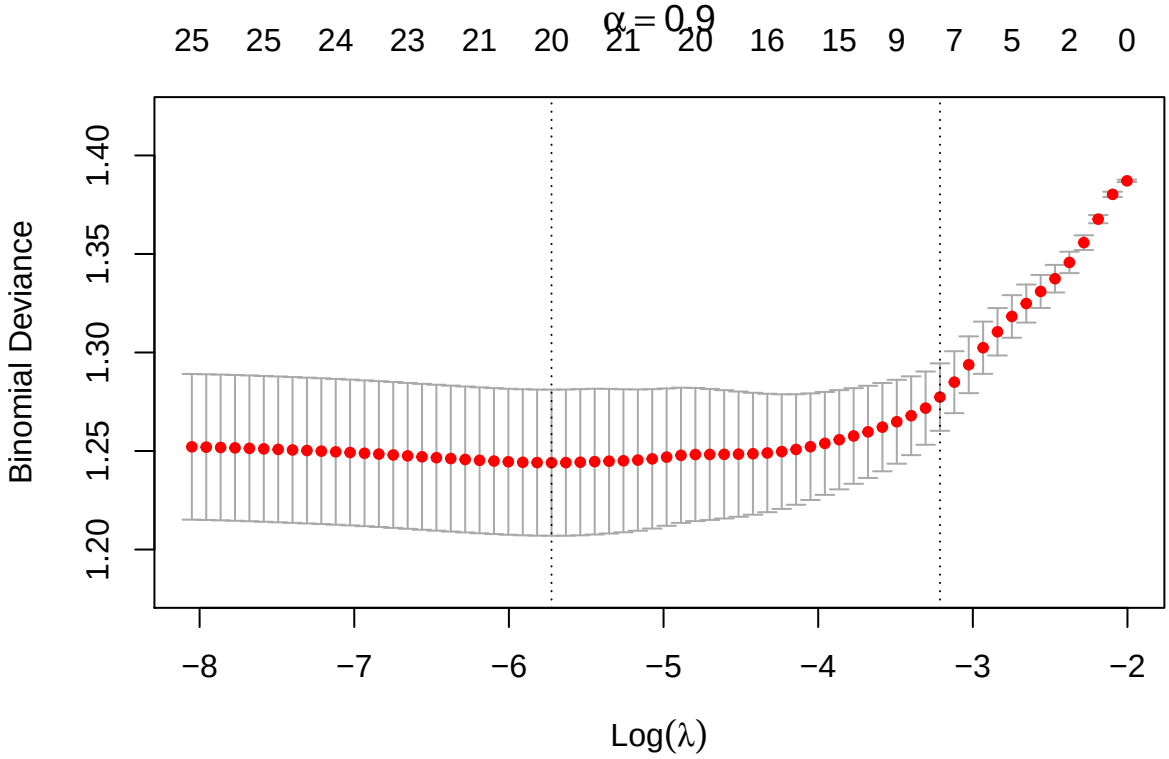

α = 0.4
25 25 25 23 22 21 22 21 20 17 14 8 7 4 2
Binomial Deviance
1.20 1.25 1.30 1.35 1.40
−7 −6 −5 −4 −3 −2 −1
Log(λ)


α = 0.5
25 25 24 23 21 21 22 21 18 15 12 7 6 2 2
Binomial Deviance
1.20 1.25 1.30 1.35 1.40
−7 −6 −5 −4 −3 −2
Log(λ)


α = 0.6
25 25 25 23 22 21 20 21 20 16 14 8 7 4 2
Binomial Deviance
1.20 1.25 1.30 1.35 1.40
−8 −7 −6 −5 −4 −3 −2
Log(λ)


α = 0.7
25 25 25 23 21 21 22 21 18 15 13 7 6 2 2
Binomial Deviance
1.20 1.25 1.30 1.35 1.40
−8 −7 −6 −5 −4 −3 −2
Log(λ)






```
> #Résumé: lambda optimal pour chaque valeur d'alpha
> layout (1)
> lapply(cv.glmn3, function(x) c(x$cvm[x$lambda == x$lambda.min],
+ + x$cvsd[x$lambda == x$lambda.min]))

$`0.1`
```

```
[1] 1.24334596 0.03240747

$`0.2`
[1] 1.24380465 0.03446807

$`0.3`
[1] 1.24387059 0.03614771

$`0.4`
[1] 1.24390553 0.03652575

$`0.5`
[1] 1.24393115 0.03668735

$`0.6`
[1] 1.24394632 0.03684874

$`0.7`
[1] 1.24399564 0.03693823

$`0.8`
[1] 1.24402662 0.03701833

$`0.9`
[1] 1.24406738 0.03707612
```

```
> #Régression logistique avec régularisation elastic-net (lambda et alpha choisis par validation croisé
> glmn3 <- glmnet(x = TRAIN.df[,var.model] %>% as.matrix,
+           y = TRAIN.df[,"STATUT"], alpha = 0.1, family = "binomial",
+           lambda = cv.glmn3[[9]]$lambda.min)
```

```
> #Prédiction à l'aide de la régression logistique avec régularisation elastic-net
> glmn3p <- predict(cv.glmn3[[9]], newx = TRAIN.df[,var.model] %>% as.matrix)
```

```
> #Table de classification montrant la performance prédictive du modèle
> cv4 <- table(1*(glmn3p>0), TRAIN.df$STATUT)
> cv4
```

```
   0   1
0 240 117
1 109 234
```

```
> prop.table(cv4)*100
```

```
          0        1
0 34.28571 16.71429
1 15.57143 33.42857
```

```
> sprintf("%.1f%% de bonne classification", sum(diag(prop.table(cv4)))*100)
```

```
[1] "67.7% de bonne classification"
```

Validation des modèles avec l'échantillon test On prend maintenant l'échantillon de test et on valide nos prédictions basées sur le modèle construit avec l'échantillon d'entraînement.

```
> #Prédiction avec les données test à partir de chaque modèle créé ci-haut
> glm1tp <- predict(glm1, newx = TEST.df[,var.model] %>% as.matrix, s = "lambda.min")
> glmn1tp <- predict(cv.glmn1, newx = TEST.df[,var.model] %>% as.matrix, s = "lambda.min")
> glmn2tp <- predict(cv.glmn2, newx = TEST.df[,var.model] %>% as.matrix, s = "lambda.min")
> glmn3tp <- predict(cv.glmn3[[9]], newx = TEST.df[,var.model] %>%as.matrix, s = "lambda.min")
```

```
> #Table de classification résumant la performance du modèle additif sur les données test
> cvt1 <- table(1*(glm1tp>0), TEST.df$STATUT)
> cvt1
```

```
     0   1
  0 100  49
  1  51 100
```

```
> prop.table(cvt1)*100
```

```
          0        1
  0 33.33333 16.33333
  1 17.00000 33.33333
```

```
> sprintf("%.1f%% de bonne classification", sum(diag(prop.table(cvt1)))*100)
```

```
[1] "66.7% de bonne classification"
```

```
> #Table de classification résumant la performance du modèle avec régularisation ridge sur les données
> cvt2 <- table(1*(glm1tp>0), TEST.df$STATUT)
> cvt2
```

```
     0   1
  0 100  49
  1  51 100
```

```
> prop.table(cvt2)*100
```

```
          0        1
  0 33.33333 16.33333
  1 17.00000 33.33333
```

```
> sprintf("%.1f%% de bonne classification", sum(diag(prop.table(cvt2)))*100)
```

```
[1] "66.7% de bonne classification"
```

```
> #Table de classification résumant la performance du modèle avec régularisation lasso sur les données
> cvt3 <- table(1*(glmn2tp>0), TEST.df$STATUT)
> cvt3
```

```
   0   1
0 101  49
1  50 100
```

```
> prop.table(cvt3)*100
```

```
          0        1
0 33.66667 16.33333
1 16.66667 33.33333
```

```
> sprintf("%.1f%% de bonne classification", sum(diag(prop.table(cvt3)))*100)
```

```
[1] "67.0% de bonne classification"
```

```
> #Table de classification résumant la performance du modèle avec régularisation elastic-net sur les do
> cvt4 <- table(1*(glmn3tp>0), TEST.df$STATUT)
> cvt4
```

```
   0   1
0 101  49
1  50 100
```

```
> prop.table(cvt4)*100
```

```
          0        1
0 33.66667 16.33333
1 16.66667 33.33333
```

```
> sprintf("%.1f%% de bonne classification", sum(diag(prop.table(cvt4)))*100)
```

```
[1] "67.0% de bonne classification"
```

```
> #Bootstrap
> set.seed(1234)
> good.class <- function(model, i ){
+             if("glm" %in% class(model)){
```

```
+             glm1tp <- predict(glm1, newdata = TEST.df[i,])
+             cvt1 <- table(1*(glm1tp>0), TEST.df$STATUT[i])
+             sum(diag(prop.table(cvt1)))*100
+ }
+   else {glmn3tp <- predict(model,
+                            newx = TEST.df[i,var.model] %>% as.matrix, s = "lambda.min")
+   (cvt4 <- table(1*(glmn3tp>0), TEST.df$STATUT)) %>%
+ sum(diag(prop.table(cvt4)))*100
+   }
+ }
>
> sd(replicate(1000,good.class(glm1, sample(1:300, 300, TRUE))))
```

```
[1] 2.860233
```

```
> #Tableau de comparaison des coefficients de chaque modèle
> length(drop(coef(cv.glmn3[[6]],
+        s = "lambda.min",allCoef = TRUE)))
```

```
[1] 26
```

```
> coef(cv.glmn3[[6]], s = "lambda.min",allCoef = TRUE)
```

```
26 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept)    0.01013546
MALE           0.08685561
AGE            0.32715857
PAR_IMM        .
MINORITY       .
SCOLMAX        0.26739839
TRAVAILM       0.10375917
TRAVAILP      -0.06856472
PAR_SEP        0.50403987
ADAPT         -0.10908915
SRDQ          -0.01417881
EVDISTSEV      0.15589798
EVDISTMOD     -0.30191322
SEVER03DICO    0.32154633
SEVER36DICO    .
SEVER69DICO   -0.29121185
SEVER912DICO   .
MODER203DICO  -0.22970279
MODER236DICO   0.19622058
MODER269DICO   0.22864685
MODER2912DICO  .
LOW203DICO     0.23449993
LOW236DICO     0.10313554
LOW269DICO    -0.21882702
LOW2912DICO   -0.13987804
CHRONSEVACT    0.55478994
```

```
> cf <- data.frame(VAR = c("Int.", var.model),
+                  OLS = drop(coef(glm1, s = "lambda.min",allCoef = TRUE)),
+                  RIDGE = drop(coef(cv.glmn1,s = "lambda.min",allCoef = TRUE)),
+                  LASSO = drop(coef(cv.glmn2,s = "lambda.min",allCoef = TRUE)),
+                  `ELASTIC NET` = drop(coef(cv.glmn3[[6]],
+                                            s = "lambda.min",allCoef = TRUE)))
> kable(cf, digits = 2,row.names = FALSE)
```

VAR	OLS	RIDGE	LASSO	ELASTIC.NET
Int.	0.01	0.01	0.01	0.01
MALE	0.12	0.09	0.09	0.09
AGE	0.38	0.26	0.33	0.33
PAR_IMM	-0.02	0.01	0.00	0.00
MINORITY	0.01	0.00	0.00	0.00
SCOLMAX	0.28	0.24	0.27	0.27
TRAVAILM	0.12	0.10	0.10	0.10
TRAVAILP	-0.11	-0.07	-0.07	-0.07
PAR_SEP	0.55	0.45	0.51	0.50
ADAPT	-0.12	-0.10	-0.11	-0.11
SRDQ	-0.04	-0.02	-0.01	-0.01
EVDISTSEV	0.26	0.10	0.16	0.16
EVDISTMOD	-0.50	-0.19	-0.31	-0.30
SEVER03DICO	0.43	0.24	0.33	0.32
SEVER36DICO	-0.06	0.05	0.00	0.00
SEVER69DICO	-0.46	-0.19	-0.30	-0.29
SEVER912DICO	-0.01	0.05	0.00	0.00
MODER203DICO	-0.37	-0.14	-0.24	-0.23
MODER236DICO	0.36	0.10	0.20	0.20
MODER269DICO	0.44	0.12	0.24	0.23
MODER2912DICO	0.10	-0.08	0.00	0.00
LOW203DICO	0.26	0.21	0.24	0.23
LOW236DICO	0.12	0.10	0.10	0.10
LOW269DICO	-0.23	-0.21	-0.22	-0.22
LOW2912DICO	-0.17	-0.13	-0.14	-0.14
CHRONSEVACT	0.61	0.46	0.56	0.55

Résumé de la performance des différents modèles:

```
Modèle additif de régression linéaire classique:
  Performance de 70.1% avec les données d'entrainement
  Performance de 66.7% avec les données de test
Régression linéaire avec régularisation ridge:
  Performance de 70.6% avec les données d'entrainement
  Performance de 66.7% avec les données de test
Régression linéaire avec régularisation lasso:
  Performance de 69.9% avec les données d'entrainement
  Performance de 67% avec les données de test
Régression linéaire avec régularisation elastic-net:
  Performance de 67.7% avec les données d'entrainement
  Performance de 67% avec les données de test
```