Projet Modèles Linéaires et ses Généralisations

Ismaël Bendib – Paul Caillere – Adrien Passuello – Axel Sauvaget

2022 - 2023

Contents

Importation des packages

```
library(MASS);library(knitr);library(ggplot2);library(cowplot);library(reshape2)
library(dplyr);library(GGally);library(corrplot);library(questionr);library(multcomp)
library(TeachingDemos);library(leaps);library(dplyr);library(ROCR);library(DAAG)
library(car)
```

Partie I - Analyse empirique descriptive des données

```
train <- read.csv(file="diabetes_train.csv", header=TRUE)
test <- read.csv(file="diabetes_test.csv", header=TRUE)
str(train)
```

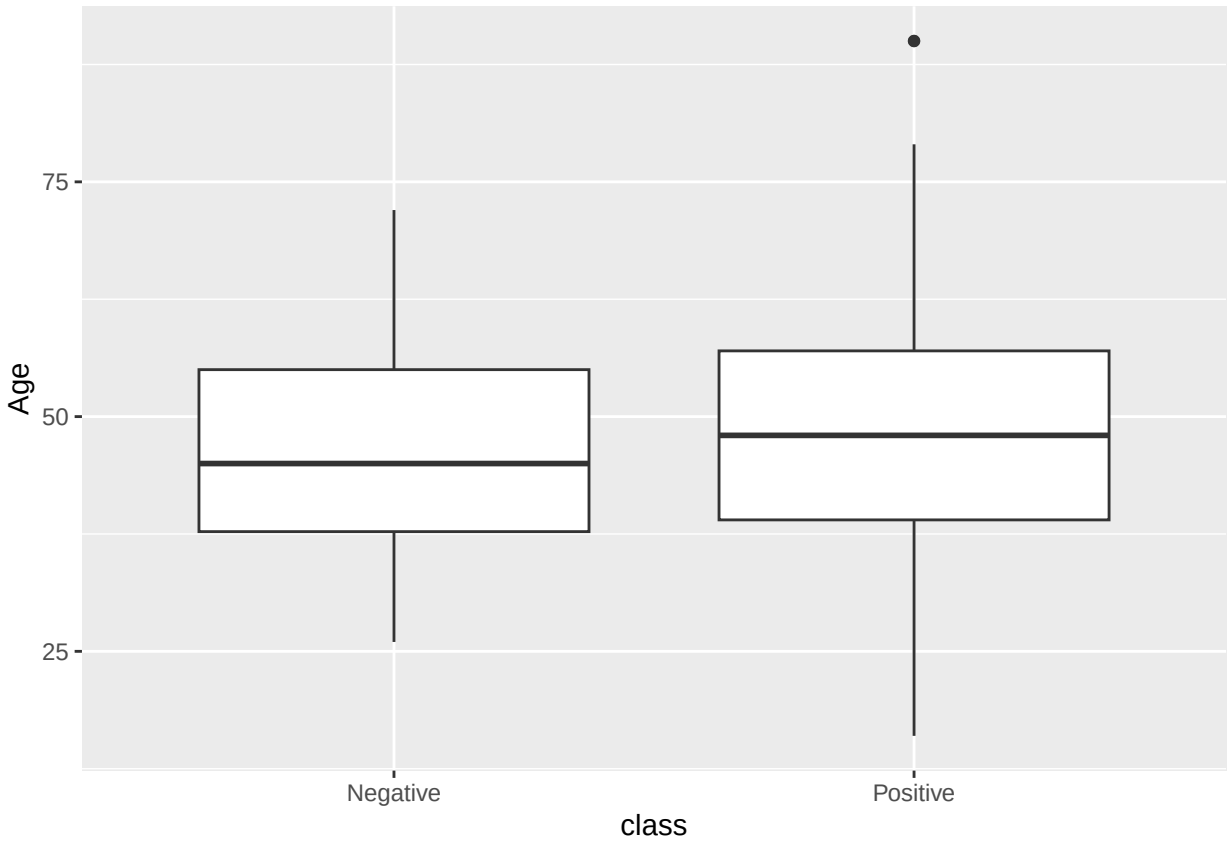
Importation des données

```
## 'data.frame':    416 obs. of  17 variables:
##  $ Age               : int  40 58 53 46 63 38 25 38 54 66 ...
##  $ Gender            : chr  "Female" "Female" "Male" "Male" ...
##  $ Polyuria          : chr  "Yes" "Yes" "No" "No" ...
##  $ Polydipsia        : chr  "Yes" "No" "No" "No" ...
##  $ sudden.weight.loss: chr  "Yes" "Yes" "No" "No" ...
##  $ weakness          : chr  "Yes" "No" "Yes" "Yes" ...
##  $ Polyphagia        : chr  "No" "Yes" "Yes" "No" ...
##  $ Genital.thrush    : chr  "No" "No" "No" "No" ...
##  $ visual.blurring   : chr  "Yes" "No" "Yes" "No" ...
##  $ Itching           : chr  "Yes" "No" "Yes" "Yes" ...
##  $ Irritability      : chr  "No" "Yes" "No" "No" ...
##  $ delayed.healing   : chr  "No" "No" "Yes" "Yes" ...
##  $ partial.paresis   : chr  "Yes" "No" "Yes" "No" ...
##  $ muscle.stiffness  : chr  "Yes" "Yes" "Yes" "No" ...
##  $ Alopecia          : chr  "No" "No" "Yes" "Yes" ...
##  $ Obesity           : chr  "No" "Yes" "No" "No" ...
##  $ class             : chr  "Positive" "Positive" "Negative" "Negative" ...
```

On cherche à expliquer le facteur `class`. Pour pouvoir l'utiliser dans un modèle linéaire généralisé (`glm`), il faut d'abord convertir les valeurs du facteur `class` en 0 (`Negative`) et 1 (`Positive`).

Mais avant cela, faisons un boxplot entre `class` et `Age` pour voir s'il y a un potentiel lien entre l'âge des patients et le diagnotic.

```
ggplot(train, aes(x=class,y=Age)) + geom_boxplot()
```



D'après ces deux boxplots, il ne semble pas y avoir de lien entre l'âge des individus et leur diagnostic.

Convertissons désormais les valeurs des tableaux de données `train` et `test` :

```
train$class<-train$class=='Positive'
test$class<-test$class=='Positive'
train$class<-as.numeric(train$class)
test$class<-as.numeric(test$class)
```

Maintenant dans les deux DataFrames, pour la variable réponse `class`, 1 correspond à `Positive` et 0 à `Negative`.

De la même façon, on convertit le facteur `Gender` : 0 pour les femmes, 1 pour les hommes.

```
train$Gender<-train$Gender=='Male'
test$Gender<-test$Gender=='Male'
train$Gender<-as.numeric(train$Gender)
test$Gender<-as.numeric(test$Gender)
```

Enfin, on transforme toutes les autres variables '`Yes`' ou '`No`' par des `1` ou `0` pour pouvoir faire une analyse empirique.

```
for(i in 1:ncol(train)){
  train[,i][train[,i] == "Yes"] <- 1
  train[, i][train[, i] == "No"] <- 0
  train[,i] <- as.numeric(train[, i])
  test[,i][test[,i] == "Yes"] <- 1
```

```
  test[, i][test[, i] == "No"] <- 0
  test[,i] <- as.numeric(test[, i])
}
```

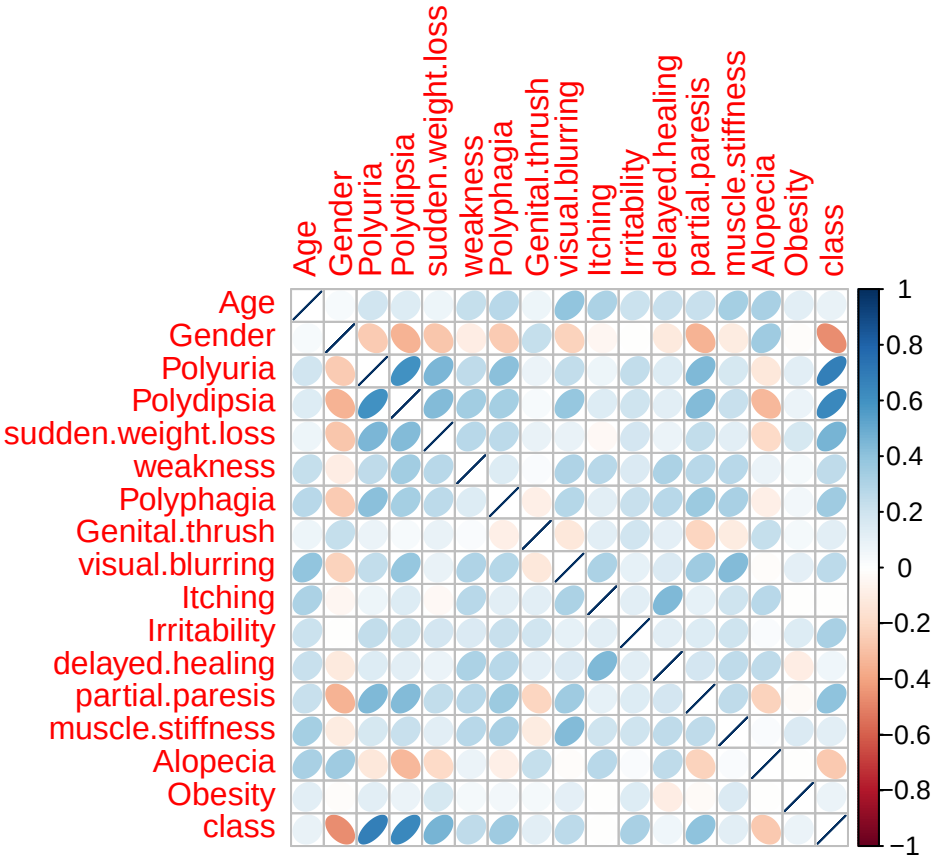
Voici le résumé des variables du nouveau tableau de données `train` :

```
str(train)
```

```
## 'data.frame':    416 obs. of  17 variables:
##  $ Age               : num  40 58 53 46 63 38 25 38 54 66 ...
##  $ Gender            : num  0 0 1 1 1 0 0 1 1 1 ...
##  $ Polyuria          : num  1 1 0 0 1 1 0 0 1 0 ...
##  $ Polydipsia        : num  1 0 0 0 1 1 0 0 1 0 ...
##  $ sudden.weight.loss: num  1 1 0 0 1 1 0 0 1 0 ...
##  $ weakness          : num  1 0 1 1 1 1 1 0 1 1 ...
##  $ Polyphagia        : num  0 1 1 0 1 1 1 0 0 0 ...
##  $ Genital.thrush    : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ visual.blurring   : num  1 0 1 0 1 1 1 0 1 1 ...
##  $ Itching           : num  1 0 1 1 0 1 0 0 1 0 ...
##  $ Irritability      : num  0 1 0 0 0 1 0 0 1 1 ...
##  $ delayed.healing   : num  0 0 1 1 0 1 0 0 1 1 ...
##  $ partial.paresis   : num  1 0 1 0 0 1 0 0 1 1 ...
##  $ muscle.stiffness  : num  1 1 1 0 1 1 0 0 1 1 ...
##  $ Alopecia          : num  0 0 1 1 1 0 1 0 0 1 ...
##  $ Obesity           : num  0 1 0 0 1 0 0 0 0 0 ...
##  $ class             : num  1 1 0 0 1 1 1 0 1 1 ...
```

Analyse des corrélations : On va analyser la corrélation entre les variables :

```
corrplot(round(cor(train),2),method="ellipse")
```



D'après cette matrice de corrélation entre les variables, les facteurs `Polyuria` et `Polydipsia` semblent fortement corrélés entre-eux et avec `class`. De même, le genre (`Gender`) semble aussi négativement corrélé avec la variable réponse `class`.

Partie II - Sélection et validation du modèle, étude d'outliers, interprétation de l'effet des covariables sur la variable d'intérêt

Puisque la variable réponse `class` ne prend que 2 valeurs possibles (0 et 1), notre modèle sera une régression logistique.

Créons un premier modèle (complet) qui prend en compte toutes les variables pour expliquer `class` :

```
mod_complet <- glm(class~., data=train, family='binomial')
coef(mod_complet)
```

```
##        (Intercept)                Age             Gender           Polyuria
##         3.99461050        -0.07875796        -4.84121687         4.76541325
##         Polydipsia sudden.weight.loss           weakness         Polyphagia
##         5.36546967         0.01930335         0.94487831         0.83036637
##     Genital.thrush    visual.blurring            Itching       Irritability
##         1.94700049         1.17218825        -3.08876953         2.90473076
##    delayed.healing    partial.paresis   muscle.stiffness           Alopecia
##        -0.36838005         0.97379106        -0.76763308         0.78715178
##             Obesity
##        -0.23132834
```

Ce modèle de régression logistique est de la forme :

$$\log\left(\frac{\widehat{p}}{1-\widehat{p}}\right) = 3.99461050 - 0.07875796 \times \texttt{Age} - 4.84121687 \times \mathbf{1}_{\texttt{Gender = Male}} + \cdots - 0.23132834 \times \mathbf{1}_{\texttt{Obesity = Yes}}$$

```
summary(mod_complet)
```

```
##
## Call:
## glm(formula = class ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.69098  -0.18946   0.00246   0.04600   2.67460
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         3.99461    1.33470   2.993 0.002764 **
## Age                -0.07876    0.03130  -2.517 0.011851 *
## Gender             -4.84122    0.71345  -6.786 1.16e-11 ***
## Polyuria            4.76541    0.83447   5.711 1.12e-08 ***
## Polydipsia          5.36547    1.04592   5.130 2.90e-07 ***
## sudden.weight.loss  0.01930    0.61758   0.031 0.975065
## weakness            0.94488    0.60415   1.564 0.117820
## Polyphagia          0.83037    0.62553   1.327 0.184357
## Genital.thrush      1.94700    0.65961   2.952 0.003160 **
## visual.blurring     1.17219    0.77096   1.520 0.128402
## Itching            -3.08877    0.82797  -3.731 0.000191 ***
## Irritability        2.90473    0.73968   3.927 8.60e-05 ***
## delayed.healing    -0.36838    0.63480  -0.580 0.561704
## partial.paresis     0.97379    0.59743   1.630 0.103106
## muscle.stiffness   -0.76763    0.68223  -1.125 0.260514
## Alopecia            0.78715    0.75420   1.044 0.296628
## Obesity            -0.23133    0.64114  -0.361 0.718245
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 554.34  on 415  degrees of freedom
## Residual deviance: 129.10  on 399  degrees of freedom
## AIC: 163.1
##
## Number of Fisher Scoring iterations: 8
```

D'après le `summary`, seules les variables `Age`, `Gender`, `Polyuria`, `Polydipsia`, `Genital.thrush`, `Itching` et `Irritability` sont significatives. C'est un résultat à prendre avec des pincettes, car le test effectué implementé pour chaque variable j est un test de Wald défini par :

$$\mathcal{H}_0 \colon \beta_j = 0 \qquad VS \qquad \mathcal{H}_1 \colon \beta_j \neq 0$$

Donc sous $\mathcal{H}_0$, on a $Z_j = \dfrac{\widehat{\beta}_j}{\widehat{\sigma}_j} \sim \mathcal{N}(0,1)$

Il suffit donc simplement de comparer Z_j au quantile d'ordre 0,95 de la loi normale centrée réduite. Si $|Z_j| > q_{0,95}^{\mathcal{N}(0,1)}$, alors on est dans la zone de rejet de $\mathcal{H}_0$.

Cependant peut-être que les variables que l'on n'a pas retenues sont significatives lorsqu'elles sont associées à un découpage différent. De plus, lorsqu'on traite des variables qualitatives, il est préférable de tester si une variable est significative ou non plutôt que de réaliser une approche coefficient par coefficient.

On va maintenant enlever des variables par différentes méthodes. Pour cela, on va effectuer plusieurs test (`anova, Anova, Backward, Forward, Both` avec les critères `AIC` par exemple). Si l'on constate que des variables ne sont
retenues dans aucun des cas, alors on les enlèvera de notre modèle.

Méthode 1 - Forward On part du modèle réduit à l'intercept (`mod0`). On compare `mod0` à tous les modèles contenant en plus 1 variable explicative. On choisit le meilleur modèle selon le critère. Puis, on ajoute 1 variable parmi les $p-1$ restantes ($p = 16$) et on choisit le meilleur modèle selon le critère et on réitère ce procédé. On s'arrête quand ajouter une variable n'améliore pas le critère.

```
mod0<-glm(class~1, data=train, family='binomial')
modForw=step(mod0,class ~ Age + Gender + Polyuria + Polydipsia + sudden.weight.loss +
    weakness + Polyphagia + Genital.thrush + visual.blurring +
    Itching + Irritability + delayed.healing + partial.paresis +
    muscle.stiffness + Alopecia + Obesity,
    trace=F,direction = c('forward'))
modForw
```

```
##
## Call:  glm(formula = class ~ Polyuria + Gender + Polydipsia + Irritability +
##     Itching + Genital.thrush + Age + visual.blurring + weakness +
##     partial.paresis, family = "binomial", data = train)
##
## Coefficients:
##     (Intercept)         Polyuria           Gender       Polydipsia
##         3.54966          4.68151         -4.70175          4.95154
##    Irritability          Itching   Genital.thrush              Age
##         2.72238         -2.90858          1.99039         -0.06701
## visual.blurring         weakness  partial.paresis
##         1.19770          0.86489          0.94325
##
## Degrees of Freedom: 415 Total (i.e. Null);  405 Residual
## Null Deviance:       554.3
## Residual Deviance: 133   AIC: 155
```

Ici, le modèle retenu est : `class ~Polyuria + Gender + Polydipsia + Irritability + Itching + Genital.thrush + Age + visual.blurring + weakness + partial.paresis`

Méthode 2 - Backward Même stratégie mais en partant du modèle complet et on enlève 1 à 1 les variables en comparant les modèles 2 à 2 selon un critère.

```
modBack=step(mod_complet,class~.,trace=F,direction = c('backward'))
modBack
```

```
##
```

```
## Call:  glm(formula = class ~ Age + Gender + Polyuria + Polydipsia +
##     weakness + Genital.thrush + visual.blurring + Itching + Irritability +
##     partial.paresis, family = "binomial", data = train)
##
## Coefficients:
##     (Intercept)              Age           Gender         Polyuria
##         3.54966         -0.06701         -4.70175          4.68151
##      Polydipsia         weakness   Genital.thrush  visual.blurring
##         4.95154          0.86489          1.99039          1.19770
##         Itching     Irritability  partial.paresis
##        -2.90858          2.72238          0.94325
##
## Degrees of Freedom: 415 Total (i.e. Null);  405 Residual
## Null Deviance:       554.3
## Residual Deviance: 133   AIC: 155
```

Ici, le modèle retenu est le suivant : `class ~Age + Gender + Polyuria + Polydipsia + weakness + Genital.thrush + visual.blurring + Itching + Irritability + partial.paresis`

Méthode 3 - Both Mixte des 2 méthodes. On part de l'intercept et on ajoute/enlève les variables 1 à 1 et on compare selon le critère.

```
modBoth=step(mod0,class ~ Age + Gender + Polyuria + Polydipsia + sudden.weight.loss +
    weakness + Polyphagia + Genital.thrush + visual.blurring +
    Itching + Irritability + delayed.healing + partial.paresis +
    muscle.stiffness + Alopecia + Obesity,
             trace=F,direction = c('both'))
modBoth
```

```
##
## Call:  glm(formula = class ~ Polyuria + Gender + Polydipsia + Irritability +
##     Itching + Genital.thrush + Age + visual.blurring + weakness +
##     partial.paresis, family = "binomial", data = train)
##
## Coefficients:
##     (Intercept)         Polyuria           Gender       Polydipsia
##         3.54966          4.68151         -4.70175          4.95154
##    Irritability          Itching   Genital.thrush              Age
##         2.72238         -2.90858          1.99039         -0.06701
## visual.blurring         weakness  partial.paresis
##         1.19770          0.86489          0.94325
##
## Degrees of Freedom: 415 Total (i.e. Null);  405 Residual
## Null Deviance:       554.3
## Residual Deviance: 133   AIC: 155
```

Ici, le modèle retenu est : `class ~Polyuria + Gender + Polydipsia + Irritability + Itching + Genital.thrush + Age + visual.blurring + weakness + partial.paresis`

Étant donné que les trois méthodes nous amènent à sélectionner le même modèle et que les `AIC` sont les mêmes, on va sélectionner ce modèle et on va lui ajouter des interactions qui semblent être pertinentes comme entre `Polyuria` (importante quantité d'urine par jour) et `Polydipsia` (sensation de soif intense et permanente), l'`Age` et `visual.blurring` (vision floue) et surtout `weakness` (faiblesses) avec toutes les variables liées à des maladies en général.

```
mod<-glm(class ~ Gender + Irritability + Age*visual.blurring + (Polyuria * Polydipsia  +
    Itching + Genital.thrush + visual.blurring +
    partial.paresis) * weakness, data=train, family='binomial')
```

```
## Warning: glm.fit: des probabilités ont été ajustées numériquement à 0 ou 1
```

```
summary(mod)
```

```
##
## Call:
## glm(formula = class ~ Gender + Irritability + Age * visual.blurring +
##     (Polyuria * Polydipsia + Itching + Genital.thrush + visual.blurring +
##         partial.paresis) * weakness, family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.71362  -0.05618   0.00000   0.00015   2.22928
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     2.01773    1.65503   1.219 0.222788
## Gender                         -6.44539    1.16900  -5.514 3.52e-08 ***
## Irritability                    5.03129    1.17717   4.274 1.92e-05 ***
## Age                            -0.05057    0.04355  -1.161 0.245528
## visual.blurring                11.89188    4.03528   2.947 0.003209 **
## Polyuria                        8.45340    1.83470   4.608 4.08e-06 ***
## Polydipsia                      5.69219    1.73888   3.273 0.001062 **
## Itching                        -3.32170    1.48925  -2.230 0.025718 *
## Genital.thrush                  3.81861    1.40347   2.721 0.006512 **
## partial.paresis                -0.47311    1.23866  -0.382 0.702495
## weakness                        5.07564    1.40088   3.623 0.000291 ***
## Age:visual.blurring            -0.14421    0.06353  -2.270 0.023211 *
## Polyuria:Polydipsia            13.57599 3636.28745   0.004 0.997021
## Polyuria:weakness              -2.92999    1.74611  -1.678 0.093346 .
## Polydipsia:weakness             0.40188    1.93536   0.208 0.835502
## Itching:weakness               -2.75707    1.82611  -1.510 0.131092
## Genital.thrush:weakness        -3.23564    1.64213  -1.970 0.048794 *
## visual.blurring:weakness       -4.00812    2.15330  -1.861 0.062690 .
## partial.paresis:weakness        0.84096    1.48105   0.568 0.570163
## Polyuria:Polydipsia:weakness    0.94919 4069.92825   0.000 0.999814
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 554.34  on 415  degrees of freedom
## Residual deviance: 101.26  on 396  degrees of freedom
## AIC: 141.26
##
## Number of Fisher Scoring iterations: 20
```

On voit avec le `summary` qu'un bon nombre d'interactions qui semblaient être pertinentes, ne le sont finalement pas. Seules les interactions `Genital.thrush:weakness`, `Age:visual.blurring` et

`visual.blurring:weakness` semblent être significatives. L'`Age` ne semble pas être une variable significative (ce qui est logique d'après le boxplot de la partie I) mais son interaction avec `visual.blurring` l'est. On garde donc uniquement l'intéraction dans notre modèle. On peut également noter que la variable `partial.paresis` n'est pas significative.

```
mod1=glm(class ~ Gender + Irritability + Age:visual.blurring + Polyuria + Polydipsia  +
    Itching + (Genital.thrush + visual.blurring) * weakness, data=train, family='binomial')
summary(mod1)
```

```
##
## Call:
## glm(formula = class ~ Gender + Irritability + Age:visual.blurring +
##     Polyuria + Polydipsia + Itching + (Genital.thrush + visual.blurring) *
##     weakness, family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.10575  -0.09583   0.00066   0.02315   2.48335
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 0.3018     0.5346   0.565 0.572316
## Gender                     -5.6830     0.9857  -5.766 8.13e-09 ***
## Irritability                3.8846     0.8863   4.383 1.17e-05 ***
## Polyuria                    6.5024     1.0866   5.984 2.18e-09 ***
## Polydipsia                  5.7427     1.1071   5.187 2.13e-07 ***
## Itching                    -4.8915     1.1153  -4.386 1.15e-05 ***
## Genital.thrush              4.0094     1.1031   3.635 0.000278 ***
## visual.blurring            13.6950     3.1840   4.301 1.70e-05 ***
## weakness                    3.5983     1.1020   3.265 0.001094 **
## Age:visual.blurring        -0.1776     0.0456  -3.894 9.86e-05 ***
## Genital.thrush:weakness    -3.7983     1.3957  -2.721 0.006501 **
## visual.blurring:weakness   -4.6505     1.4246  -3.265 0.001096 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 554.34  on 415  degrees of freedom
## Residual deviance: 112.33  on 404  degrees of freedom
## AIC: 136.33
##
## Number of Fisher Scoring iterations: 9
```

On a drastiquement réduit la déviance du modèle Residual deviance mod_complet = 129.10 Residual deviance modf = 112.33

```
anova(mod1, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
```

```
## Response: class
##
## Terms added sequentially (first to last)
##
##
##                           Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                        415     554.34
## Gender                     1  101.494       414     452.85 < 2.2e-16 ***
## Irritability               1   56.545       413     396.30 5.493e-14 ***
## Polyuria                   1  166.738       412     229.57 < 2.2e-16 ***
## Polydipsia                 1   50.766       411     178.80 1.041e-12 ***
## Itching                    1   25.789       410     153.01 3.809e-07 ***
## Genital.thrush             1    6.422       409     146.59 0.0112685 *
## visual.blurring            1    1.847       408     144.74 0.1741659
## weakness                   1    1.797       407     142.95 0.1800515
## Age:visual.blurring        1   12.919       406     130.03 0.0003252 ***
## Genital.thrush:weakness    1    2.978       405     127.05 0.0844237 .
## visual.blurring:weakness   1   14.722       404     112.33 0.0001246 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
library(car)
Anova(mod1, type = "III", test.statistic = "LR")
```

```
## Analysis of Deviance Table (Type III tests)
##
## Response: class
##                           LR Chisq Df Pr(>Chisq)
## Gender                      84.671  1  < 2.2e-16 ***
## Irritability                29.260  1  6.329e-08 ***
## Polyuria                    97.912  1  < 2.2e-16 ***
## Polydipsia                  62.262  1  3.007e-15 ***
## Itching                     36.699  1  1.379e-09 ***
## Genital.thrush              18.124  1  2.070e-05 ***
## visual.blurring             27.674  1  1.436e-07 ***
## weakness                    15.515  1  8.187e-05 ***
## Age:visual.blurring         18.794  1  1.456e-05 ***
## Genital.thrush:weakness      8.863  1  0.0029104 **
## visual.blurring:weakness    14.722  1  0.0001246 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Enfin en réalisant un test anova de type I, on trouve que les variables `visual.blurring` et `weakness` ne sont pas significatives. Mais dans le test Anova de type III, celles-ci sont bien significatives, on va donc les conserver dans le modèle.

Voici donc le modèle final :

```
modf=glm(class ~ Gender + Irritability + Polyuria + Polydipsia  +
    Itching + (Genital.thrush + visual.blurring) * weakness + Age:visual.blurring, data=train, family='
summary(modf)
```

```
##
```

```
## Call:
## glm(formula = class ~ Gender + Irritability + Polyuria + Polydipsia +
##     Itching + (Genital.thrush + visual.blurring) * weakness +
##     Age:visual.blurring, family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.10575  -0.09583   0.00066   0.02315   2.48335
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  0.3018     0.5346   0.565 0.572316
## Gender                      -5.6830     0.9857  -5.766 8.13e-09 ***
## Irritability                 3.8846     0.8863   4.383 1.17e-05 ***
## Polyuria                     6.5024     1.0866   5.984 2.18e-09 ***
## Polydipsia                   5.7427     1.1071   5.187 2.13e-07 ***
## Itching                     -4.8915     1.1153  -4.386 1.15e-05 ***
## Genital.thrush               4.0094     1.1031   3.635 0.000278 ***
## visual.blurring             13.6950     3.1840   4.301 1.70e-05 ***
## weakness                     3.5983     1.1020   3.265 0.001094 **
## Genital.thrush:weakness     -3.7983     1.3957  -2.721 0.006501 **
## visual.blurring:weakness    -4.6505     1.4246  -3.265 0.001096 **
## visual.blurring:Age         -0.1776     0.0456  -3.894 9.86e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 554.34  on 415  degrees of freedom
## Residual deviance: 112.33  on 404  degrees of freedom
## AIC: 136.33
##
## Number of Fisher Scoring iterations: 9
```

Partie III - Prédiction de la variable d'intérêt et évaluation du modèle sur les données

Erreur quadratique moyenne (MSE) :

Échantillon train : Dans un premier temps on va prédire avec le modèle final :

```
p_train <- predict(modf,  type='response')
```

Et ensuite avec le modèle complet qu'on a sélectionné pour pouvoir comparer les deux :

```
p_train_complet <- predict(mod_complet, type='response')
```

Comme mesure on peut utiliser la MSE (erreur quadratique moyenne), c'est l'erreur moyenne entre la prédiction et la réalité :

```
MSE <- c(sqrt(sum((train$class-p_train)**2)/length(train)),
         sqrt(sum((train$class-p_train_complet)**2)/length(train)))
names(MSE)=c("modf","mod_complet")
kable(data.frame(MSE))
```

	MSE
modf	1.029489
mod_complet	1.085621

La MSE issue du modèle complet est plus grande que celle du modèle final.

Échantillon test : Dans un premier temps on va prédire avec le modèle final :

```
p_test <- predict(modf, newdata=test,  type='response')
```

Et ensuite avec le modèle complet qu'on a sélectionné pour pouvoir comparer les deux :

```
p_test_complet <- predict(mod_complet, newdata=test, type='response')
```

Comme mesure on peut utiliser la MSE (erreur quadratique moyenne), c'est l'erreur moyenne entre la prédiction et la réalité :

```
MSE <- c(sqrt(sum((test$class-p_test)**2)/length(test)),
         sqrt(sum((test$class-p_test_complet)**2)/length(test)))
names(MSE)=c("modf","mod_complet")
kable(data.frame(MSE))
```

	MSE
modf	0.6927533
mod_complet	0.6234274

La MSE issue du modèle final est plus grande que celle du modèle complet.

Prédiction et tables de confusion :

Échantillon train : On peut regarder aussi le nombre de prédictions erronées. Regardons quand la probabilité de se tromper est de 0,5 :

```
prediction_train <- as.factor(p_train>0.5)
prediction_train <- ifelse(p_train>0.5, 1, 0)
```

Calculons la matrice de confusion associée aux prédictions de la régression logistique (pour un palier de classification fixé à 0,5) :

```
tabconfusion0_5_train <- table(train$class, prediction_train)
tabconfusion0_5_train
```

```
##    prediction_train
##       0   1
##   0 146  14
##   1  13 243
```

On voit que l'on a 27 erreurs de prédiction au total.

```
error <- mean(prediction_train!=(as.numeric(train$class)-1))
print(paste("Le taux d'erreurs de prédiction est d'environ :", 1-error))
```

```
## [1] "Le taux d'erreurs de prédiction est d'environ : 0.03125"
```

```
print(paste("Le taux de vrais positifs est ", tabconfusion0_5_train[2,2]/
              (tabconfusion0_5_train[2,1]+tabconfusion0_5_train[2,2])))
```

```
## [1] "Le taux de vrais positifs est  0.94921875"
```

```
print(paste("Le taux de faux positifs est ",tabconfusion0_5_train[1,2]/
              (tabconfusion0_5_train[1,1]+tabconfusion0_5_train[1,2])))
```

```
## [1] "Le taux de faux positifs est  0.0875"
```

```
print(paste("Le taux de vrais négatifs est ", tabconfusion0_5_train[1,1]/
              (tabconfusion0_5_train[1,1]+tabconfusion0_5_train[1,2])))
```

```
## [1] "Le taux de vrais négatifs est  0.9125"
```

```
print(paste("Le taux de faux négatifs est ",tabconfusion0_5_train[2,1]/
              (tabconfusion0_5_train[2,2]+tabconfusion0_5_train[2,1])))
```

```
## [1] "Le taux de faux négatifs est  0.05078125"
```

Le palier de classification 0,5 de la régression logistique doit être modifié pour améliorer nos résultats. Les coefficients estimés restent inchangés, tout comme les probabilités estimées d'être atteint du diabète. Réduisons donc palier de classification à 0,1 et calculons la matrice de confusion et l'erreur.

```
prediction_label_train <- as.numeric(p_train>0.1)
tabconfusion0_9_train <- table(train$class, prediction_label_train)
tabconfusion0_9_train
```

```
##    prediction_label_train
##       0   1
##   0 108  52
##   1   2 254
```

On voit que l'on a augmenté le nombre d'erreurs de prédiction à 54 au total, mais on a réduit le nombre de faux négatifs en passant de 13 à 2.

```
error <- mean(prediction_label_train!=(as.numeric(train$class)-1))
print(paste("Le taux d'erreurs de prédiction est d'environ :", 1-error))
```

```
## [1] "Le taux d'erreurs de prédiction est d'environ : 0.00480769230769229"
```

```
print(paste("Le taux de vrais positifs est ", tabconfusion0_9_train[2,2]/
              (tabconfusion0_9_train[2,1]+tabconfusion0_9_train[2,2])))
```

```
## [1] "Le taux de vrais positifs est  0.9921875"
```

```
print(paste("Le taux de faux positifs est ",tabconfusion0_9_train[1,2]/
              (tabconfusion0_9_train[1,1]+tabconfusion0_9_train[1,2])))
```

```
## [1] "Le taux de faux positifs est  0.325"
```

```
print(paste("Le taux de vrais négatifs est ", tabconfusion0_9_train[1,1]/
              (tabconfusion0_9_train[1,1]+tabconfusion0_9_train[1,2])))
```

```
## [1] "Le taux de vrais négatifs est  0.675"
```

```
print(paste("Le taux de faux négatifs est ",tabconfusion0_9_train[2,1]/
              (tabconfusion0_9_train[2,2]+tabconfusion0_9_train[2,1])))
```

```
## [1] "Le taux de faux négatifs est  0.0078125"
```

Etant donné que dans le cas du diagnostique du diabète, une erreur de type faux négatif est bien plus grave qu'une erreur de type faux positif, on privilégier la sensibilité et diminuer le seuil de classification "positif au diabète" de 0,5 à 0,1.

Notre modèle de prédiction a une bonne performance en ce qui concerne la détection du diabète. Il y a seulement 2 cas sur 104 pour lesquels il ne parvient pas à détecter la maladie, tandis qu'il prévoit 13 cas de diabète chez des personnes en réalité en bonne santé. Cependant, en supposant que le traitement du diabète n'aura pas trop d'effets négatifs sur les personnes qui ne sont pas atteintes de cette maladie, on peut être satisfait de notre modèle.

Cependant, on peut remarquer qu'avec le modèle complet on a :

```
prediction_complet_train <- as.numeric(p_train_complet>0.1)
tabconfusion_complet_0_9_train <- table(train$class, prediction_complet_train)
tabconfusion_complet_0_9_train
```

```
##    prediction_complet_train
##       0   1
##   0 121  39
##   1   4 252
```

```
error <- mean(prediction_complet_train!=(as.numeric(train$class)-1))
print(paste("Le taux d'erreurs de prédiction est d'environ :", 1-error))
```

```
## [1] "Le taux d'erreurs de prédiction est d'environ : 0.00961538461538458"
```

```
print(paste("Le taux de vrais positifs est ", tabconfusion_complet_0_9_train[2,2]/
              (tabconfusion_complet_0_9_train[2,1]+tabconfusion_complet_0_9_train[2,2])))
```

```
## [1] "Le taux de vrais positifs est  0.984375"
```

```
print(paste("Le taux de faux positifs est ",tabconfusion_complet_0_9_train[1,2]/
              (tabconfusion_complet_0_9_train[1,1]+tabconfusion_complet_0_9_train[1,2])))
```

```
## [1] "Le taux de faux positifs est  0.24375"
```

```
print(paste("Le taux de vrais négatifs est ", tabconfusion_complet_0_9_train[1,1]/
              (tabconfusion_complet_0_9_train[1,1]+tabconfusion_complet_0_9_train[1,2])))
```

```
## [1] "Le taux de vrais négatifs est  0.75625"
```

```
print(paste("Le taux de faux négatifs est ",tabconfusion_complet_0_9_train[2,1]/
              (tabconfusion_complet_0_9_train[2,2]+tabconfusion_complet_0_9_train[2,1])))
```

```
## [1] "Le taux de faux négatifs est  0.015625"
```

Avec le modèle complet, il y a toujours que 4 personnes sur 416 où l'on arrive pas à prévoir qu'ils ont le diabète. On prévoit 13 personnes de moins ayant du diabète alors qu'ils n'en ont pas.

Échantillon test : Pour l'échantillon test, regardons quand la probabilité de se tromper est de 0,5 :

```
prediction_test <- as.factor(p_test>0.5)
prediction_test <- ifelse(p_test>0.5, 1, 0)
```

Calculons la matrice de confusion associée aux prédictions de la régression logistique (pour un palier de classification fixé à 0,5) :

```
tabconfusion0_5_test <- table(test$class, prediction_test)
tabconfusion0_5_test
```

```
##    prediction_test
##      0  1
##   0 35  5
##   1  6 58
```

On voit que l'on a 11 erreurs de prédiction au total.

```
error <- mean(prediction_test!=(as.numeric(test$class)-1))
print(paste("Le taux d'erreurs de prédiction est d'environ :", 1-error))
```

```
## [1] "Le taux d'erreurs de prédiction est d'environ : 0.0576923076923077"
```

```
print(paste("Le taux de vrais positifs est ", tabconfusion0_5_test[2,2]/
              (tabconfusion0_5_test[2,1]+tabconfusion0_5_test[2,2])))
```

```
## [1] "Le taux de vrais positifs est  0.90625"
```

```
print(paste("Le taux de faux positifs est ",tabconfusion0_5_test[1,2]/
              (tabconfusion0_5_test[1,1]+tabconfusion0_5_test[1,2])))
```

```
## [1] "Le taux de faux positifs est  0.125"
```

```
print(paste("Le taux de vrais négatifs est ", tabconfusion0_5_test[1,1]/
              (tabconfusion0_5_test[1,1]+tabconfusion0_5_test[1,2])))
```

```
## [1] "Le taux de vrais négatifs est  0.875"
```

```
print(paste("Le taux de faux négatifs est ",tabconfusion0_5_test[2,1]/
              (tabconfusion0_5_test[2,2]+tabconfusion0_5_test[2,1])))
```

```
## [1] "Le taux de faux négatifs est  0.09375"
```

Le palier de classification 0,5 de la régression logistique doit être modifié pour améliorer nos résultats. Les coefficients estimés restent inchangés, tout comme les probabilités estimées d'être atteint du diabète. Réduisons donc palier de classification à 0,1 et calculons la matrice de confusion et l'erreur.

```
prediction_label_test <- as.numeric(p_test>0.1)
tabconfusion0_9_test <- table(test$class, prediction_label_test)
tabconfusion0_9_test
```

```
##    prediction_label_test
##      0  1
##   0 27 13
##   1  2 62
```

On voit que l'on a augmenté le nombre d'erreurs de prédiction à 15 au total, mais on a réduit le nombre de faux négatifs en passant de 6 à 2.

```
error <- mean(prediction_label_test!=(as.numeric(test$class)-1))
print(paste("Le taux d'erreurs de prédiction est d'environ :", 1-error))
```

```
## [1] "Le taux d'erreurs de prédiction est d'environ : 0.0192307692307693"
```

```
print(paste("Le taux de vrais positifs est ", tabconfusion0_9_test[2,2]/
              (tabconfusion0_9_test[2,1]+tabconfusion0_9_test[2,2])))
```

```
## [1] "Le taux de vrais positifs est  0.96875"
```

```
print(paste("Le taux de faux positifs est ",tabconfusion0_9_test[1,2]/
              (tabconfusion0_9_test[1,1]+tabconfusion0_9_test[1,2])))
```

```
## [1] "Le taux de faux positifs est  0.325"
```

```
print(paste("Le taux de vrais négatifs est ", tabconfusion0_9_test[1,1]/
              (tabconfusion0_9_test[1,1]+tabconfusion0_9_test[1,2])))
```

```
## [1] "Le taux de vrais négatifs est  0.675"
```

```
print(paste("Le taux de faux négatifs est ",tabconfusion0_9_test[2,1]/
              (tabconfusion0_9_test[2,2]+tabconfusion0_9_test[2,1])))
```

```
## [1] "Le taux de faux négatifs est  0.03125"
```

Etant donné que dans le cas du diagnostique du diabète, une erreur de type faux négatif est bien plus grave qu'une erreur de type faux positif, on privilégier la sensibilité et diminuer le seuil de classification "positif au diabète" de 0,5 à 0,1.

Notre modèle de prédiction a une bonne performance en ce qui concerne la détection du diabète. Il y a seulement 2 cas sur 104 pour lesquels il ne parvient pas à détecter la maladie, tandis qu'il prévoit 13 cas de diabète chez des personnes en réalité en bonne santé. Cependant, en supposant que le traitement du diabète n'aura pas trop d'effets négatifs sur les personnes qui ne sont pas atteintes de cette maladie, on peut être satisfait de notre modèle.

Cependant, on peut remarquer qu'avec le modèle complet on a :

```
prediction_complet_test <- as.numeric(p_test_complet>0.1)
tabconfusion_complet_0_9_test <- table(test$class, prediction_complet_test)
tabconfusion_complet_0_9_test
```

```
##    prediction_complet_test
##      0  1
##   0 29 11
##   1  2 62
```

```
error <- mean(prediction_label_test!=(as.numeric(test$class)-1))
print(paste("Le taux d'erreurs de prédiction est d'environ :", 1-error))
```

```
## [1] "Le taux d'erreurs de prédiction est d'environ : 0.0192307692307693"
```

```
print(paste("Le taux de vrais positifs est ", tabconfusion_complet_0_9_test[2,2]/
              (tabconfusion_complet_0_9_test[2,1]+tabconfusion_complet_0_9_test[2,2])))
```

```
## [1] "Le taux de vrais positifs est  0.96875"
```

```
print(paste("Le taux de faux positifs est ",tabconfusion_complet_0_9_test[1,2]/
              (tabconfusion_complet_0_9_test[1,1]+tabconfusion_complet_0_9_test[1,2])))
```

```
## [1] "Le taux de faux positifs est  0.275"
```

```
print(paste("Le taux de vrais négatifs est ", tabconfusion_complet_0_9_test[1,1]/
              (tabconfusion_complet_0_9_test[1,1]+tabconfusion_complet_0_9_test[1,2])))
```

```
## [1] "Le taux de vrais négatifs est  0.725"
```

```
print(paste("Le taux de faux négatifs est ",tabconfusion_complet_0_9_test[2,1]/
              (tabconfusion_complet_0_9_test[2,2]+tabconfusion_complet_0_9_test[2,1])))
```

```
## [1] "Le taux de faux négatifs est  0.03125"
```

Avec le modèle complet, il n'y a toujours que 2 personnes sur 104 où l'on arrive pas à prévoir qu'ils ont le diabète. Mais on prévoit 2 personnes de moins ayant du diabète alors qu'ils n'en ont pas. Donc en terme de prédiction, le modèle complet semble être meilleur que notre modèle.

ROC (Receiver Operating Characteristic) :

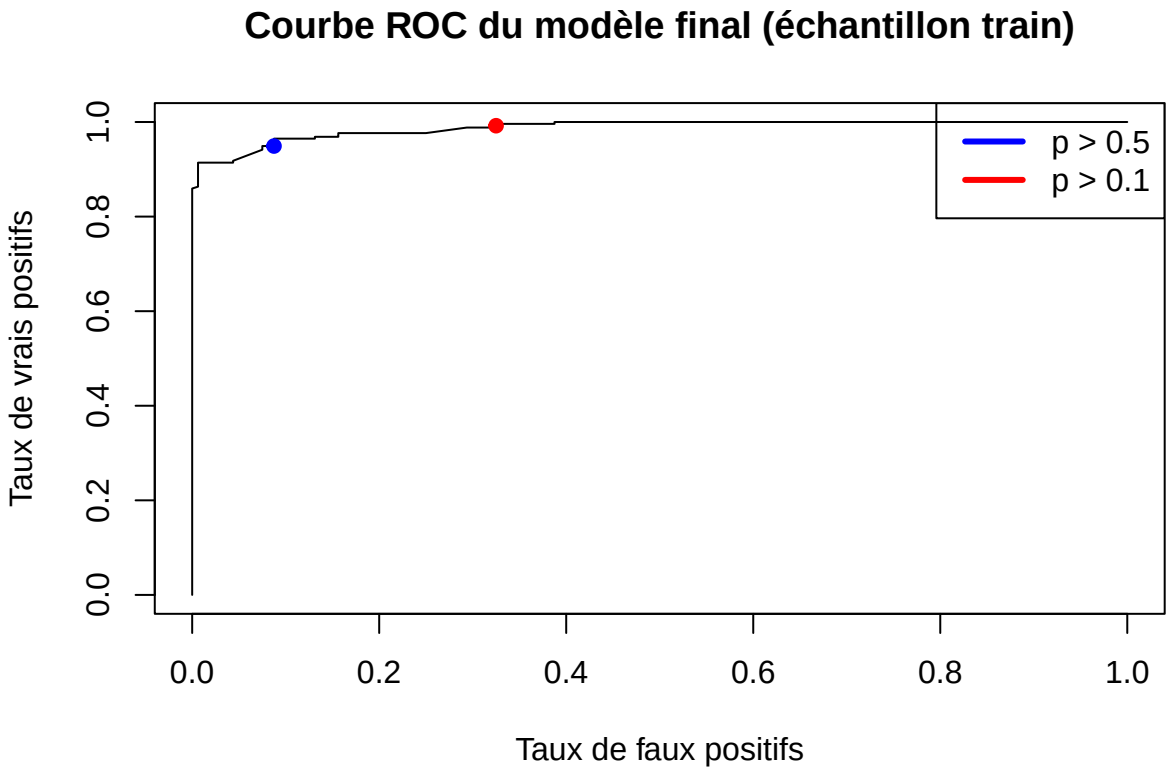
La courbe ROC est une courbe générée en représentant le taux de vrais positifs en fonction du taux de faux positifs pour des paliers de prédiction différents entre 0 et 1.

Échantillon train Pour le modèle final `modf` :

```
library(ROCR)
ROC_train <- performance(prediction(p_train, train$class), measure = "tpr",
                         x.measure = "fpr")
plot(ROC_train, colorize = FALSE, main = "Courbe ROC du modèle final (échantillon train)",
     xlab = "Taux de faux positifs", ylab = "Taux de vrais positifs")

TVP0_5_train = tabconfusion0_5_train[2,2]/(tabconfusion0_5_train[2,1] +
                                           tabconfusion0_5_train[2,2])
TFP0_5_train = tabconfusion0_5_train[1,2]/(tabconfusion0_5_train[1,1] +
                                           tabconfusion0_5_train[1,2])
TVP0_9_train = tabconfusion0_9_train[2,2]/(tabconfusion0_9_train[2,1] +
                                           tabconfusion0_9_train[2,2])
TFP0_9_train = tabconfusion0_9_train[1,2]/(tabconfusion0_9_train[1,1] +
                                           tabconfusion0_9_train[1,2])

points(x = c(TFP0_5_train, TFP0_9_train), y = c(TVP0_5_train, TVP0_9_train) ,
       col=c("blue", "red"), pch=20, lwd = 3)
legend("topright",c("p > 0.5", "p > 0.1"), col = c("blue", "red") , lwd = 3)
```



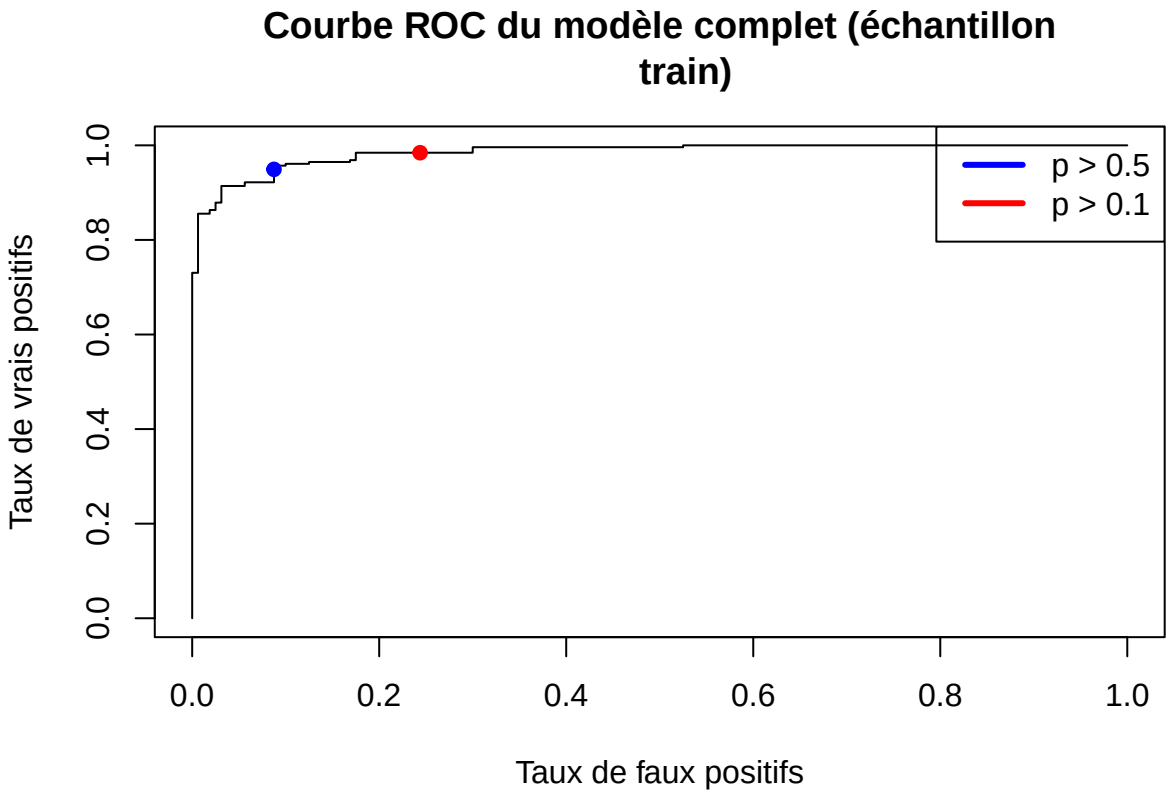
Pour le modèle complet :

```
prediction_complet_train <- as.numeric(p_train_complet>0.5)
tabconfusion_complet_0_5_train<- table(train$class, prediction_complet_train)

ROC_complet_train <- performance(prediction(p_train_complet, train$class),
                                 measure = "tpr", x.measure = "fpr")
plot(ROC_complet_train, colorize = FALSE, main = "Courbe ROC du modèle complet (échantillon
     train)", xlab = "Taux de faux positifs", ylab = "Taux de vrais positifs")


TVP0_5_train = tabconfusion_complet_0_5_train[2,2]/
  (tabconfusion_complet_0_5_train[2,1] + tabconfusion_complet_0_5_train[2,2])
TFP0_5_train = tabconfusion_complet_0_5_train[1,2]/
  (tabconfusion_complet_0_5_train[1,1] + tabconfusion_complet_0_5_train[1,2])
TVP0_9_train = tabconfusion_complet_0_9_train[2,2]/
  (tabconfusion_complet_0_9_train[2,1] + tabconfusion_complet_0_9_train[2,2])
TFP0_9_train = tabconfusion_complet_0_9_train[1,2]/
  (tabconfusion_complet_0_9_train[1,1] + tabconfusion_complet_0_9_train[1,2])


points(x = c(TFP0_5_train, TFP0_9_train), y = c(TVP0_5_train, TVP0_9_train) ,
       col=c("blue", "red"), pch=20, lwd = 3)
legend("topright",c("p > 0.5", "p > 0.1"), col = c("blue", "red") , lwd = 3)
```

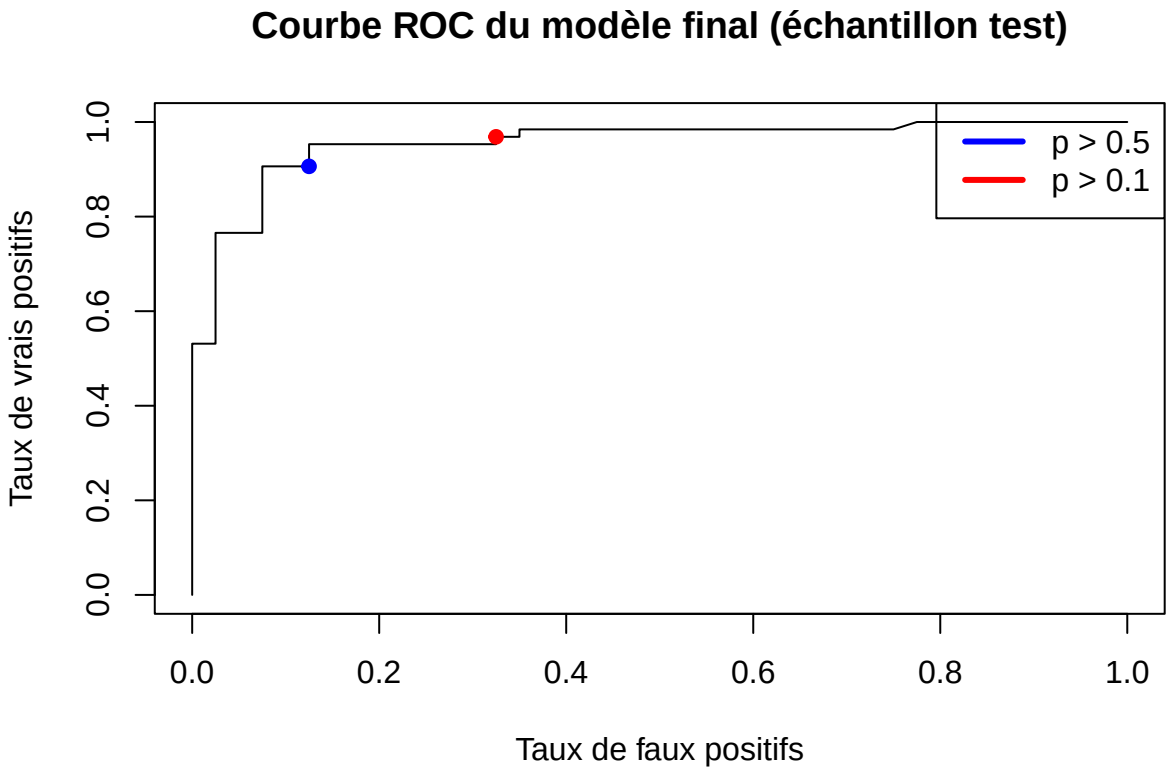


Échantillon test **Pour le modèle final `modf` :**

```
ROC_test <- performance(prediction(p_test, test$class), measure = "tpr",
                        x.measure = "fpr")
plot(ROC_test, colorize = FALSE, main = "Courbe ROC du modèle final (échantillon test)",
     xlab = "Taux de faux positifs", ylab = "Taux de vrais positifs")

TVP0_5_test = tabconfusion0_5_test[2,2]/(tabconfusion0_5_test[2,1] +
                                         tabconfusion0_5_test[2,2])
TFP0_5_test = tabconfusion0_5_test[1,2]/(tabconfusion0_5_test[1,1] +
                                         tabconfusion0_5_test[1,2])
TVP0_9_test = tabconfusion0_9_test[2,2]/(tabconfusion0_9_test[2,1] +
                                         tabconfusion0_9_test[2,2])
TFP0_9_test = tabconfusion0_9_test[1,2]/(tabconfusion0_9_test[1,1] +
                                         tabconfusion0_9_test[1,2])

points(x = c(TFP0_5_test, TFP0_9_test), y = c(TVP0_5_test, TVP0_9_test) ,
       col=c("blue", "red"), pch=20, lwd = 3)
legend("topright",c("p > 0.5", "p > 0.1"), col = c("blue", "red") , lwd = 3)
```



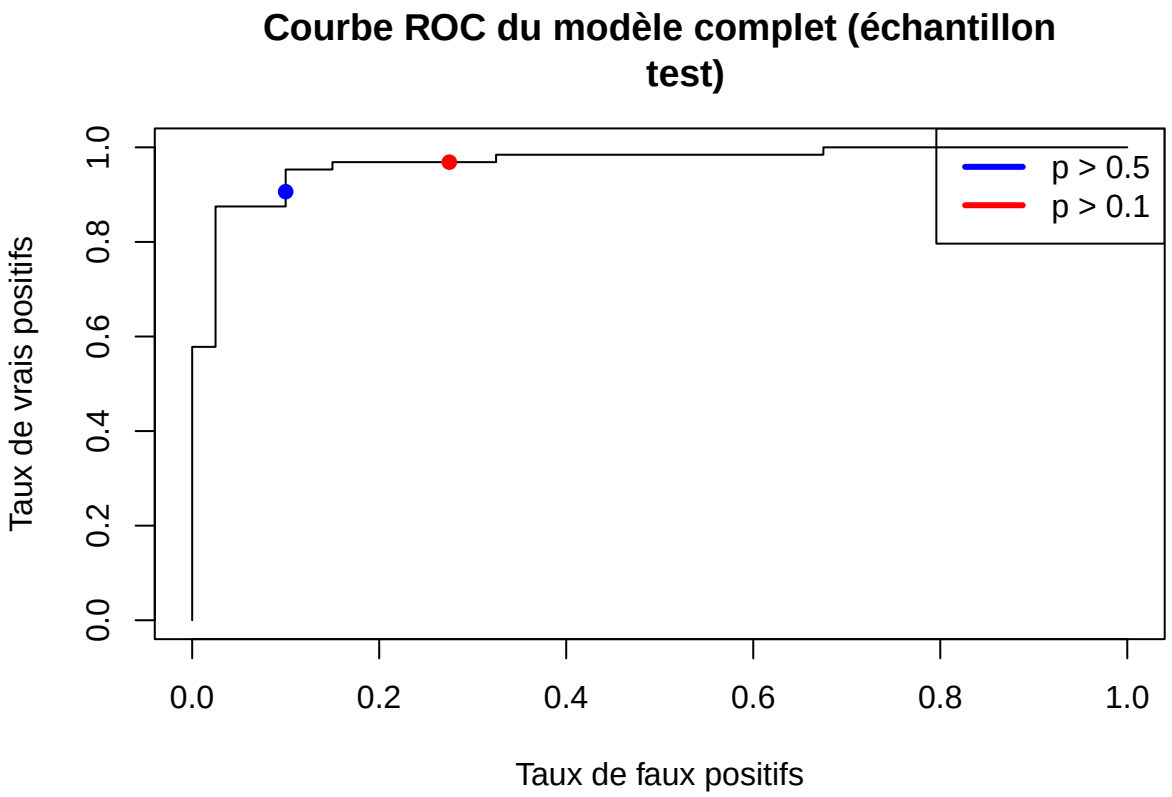
Pour le modèle complet :

```
prediction_complet_test <- as.numeric(p_test_complet>0.5)
tabconfusion_complet_0_5_test <- table(test$class, prediction_complet_test)

ROC_complet_test <- performance(prediction(p_test_complet, test$class),
                                measure = "tpr", x.measure = "fpr")
plot(ROC_complet_test, colorize = FALSE, main = "Courbe ROC du modèle complet (échantillon
     test)", xlab = "Taux de faux positifs", ylab = "Taux de vrais positifs")


TVP0_5_test = tabconfusion_complet_0_5_test[2,2]/
  (tabconfusion_complet_0_5_test[2,1] + tabconfusion_complet_0_5_test[2,2])
TFP0_5_test = tabconfusion_complet_0_5_test[1,2]/
  (tabconfusion_complet_0_5_test[1,1] + tabconfusion_complet_0_5_test[1,2])
TVP0_9_test = tabconfusion_complet_0_9_test[2,2]/
  (tabconfusion_complet_0_9_test[2,1] + tabconfusion_complet_0_9_test[2,2])
TFP0_9_test = tabconfusion_complet_0_9_test[1,2]/
  (tabconfusion_complet_0_9_test[1,1] + tabconfusion_complet_0_9_test[1,2])


points(x = c(TFP0_5_test, TFP0_9_test), y = c(TVP0_5_test, TVP0_9_test) ,
       col=c("blue", "red"), pch=20, lwd = 3)
legend("topright",c("p > 0.5", "p > 0.1"), col = c("blue", "red") , lwd = 3)
```



Nota Bene : voici également les mêmes courbes ROC (pour les deux échantillons) en utilisant la librarie `plotROC`.

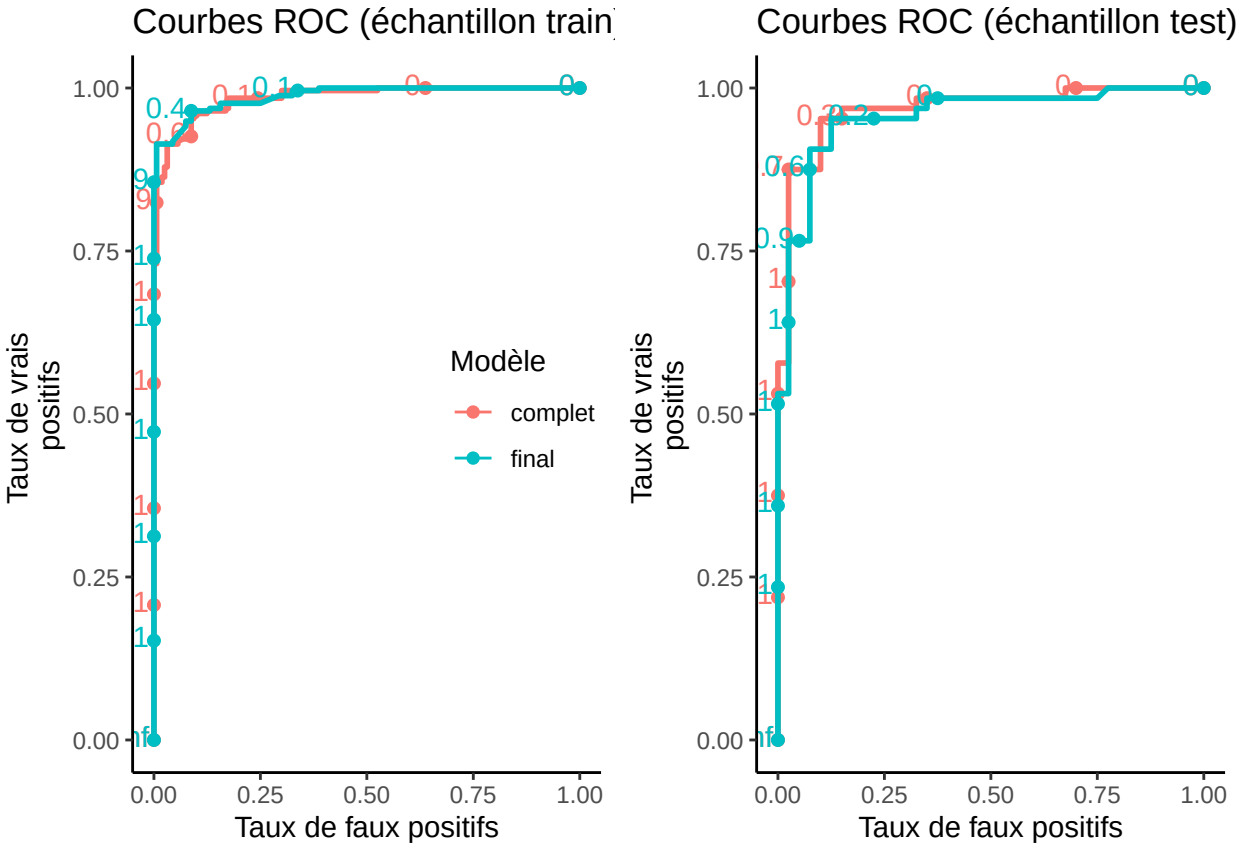
```
library(plotROC)
library(tidyr)
prev_prob_test <- data.frame(complet=predict(mod_complet,newdata=test, type="response"),
                             final=predict(modf,newdata=test,type="response"))
df_roc_test <- prev_prob_test %>% mutate(obs_test=test$class) %>%
gather(key=Modèle,value=score,complet,final)

prev_prob_train <- data.frame(complet=predict(mod_complet, type="response"),
                              final=predict(modf,type="response"))
df_roc_train <- prev_prob_train %>% mutate(obs_train=train$class) %>%
gather(key=Modèle,value=score,complet,final)

plot_ROC_train <- ggplot(df_roc_train)+aes(d=obs_train,m=score,color=Modèle)+ geom_roc()+
  theme_classic()+ggtitle("Courbes ROC (échantillon train)")+labs(y = "Taux de vrais
  positifs", x = "Taux de faux positifs")+theme(legend.position = "none")

plot_ROC_test <- ggplot(df_roc_test)+aes(d=obs_test,m=score,color=Modèle)+ geom_roc()+
  theme_classic()+ggtitle("Courbes ROC (échantillon test)")+labs(y = "Taux de vrais
  positifs", x = "Taux de faux positifs")+theme(legend.position = c(-0.5,0.5))

plot_grid(plot_ROC_train,  plot_ROC_test, ncol = 2, nrow = 1)
```



AUC (area under the curve) :

L'AUC est l'aire sous la courbe ROC. Intuitivement, un modèle avec de bonnes capacités de prédiction devrait avoir un AUC plus proche de 1 (1 étant idéal) que de 0,5. Utilisons la fonction performance pour calculer l'AUC des deux modèles.

Échantillon train : Pour le modèle final `modf` :

```
auc_train <- performance(prediction(p_train,train$class), measure='auc')
auc_train <- auc_train@y.values[[1]]
auc_train
```

```
## [1] 0.9871826
```

Pour le modèle complet :

```
auc_complet_train <- performance(prediction(p_train_complet,train$class), measure='auc')
auc_complet_train <- auc_complet_train@y.values[[1]]
auc_complet_train
```

```
## [1] 0.9842041
```

Sur l'échantillon train, les AUC montre que le modèle final est meilleur que le modèle complet mais les valeurs sont très proches.

Échantillon test : Pour le modèle final `modf` :

```
auc_test <- performance(prediction(p_test,test$class), measure='auc')
auc_test <- auc_test@y.values[[1]]
auc_test
```

```
## [1] 0.9552734
```

Pour le modèle complet :

```
auc_complet_test <- performance(prediction(p_test_complet,test$class), measure='auc')
auc_complet_test <- auc_complet_test@y.values[[1]]
auc_complet_test
```

```
## [1] 0.9667969
```

Sur l'échantillon test, les AUC montre que le modèle complet est meilleur que le modèle final mais les valeurs sont très proches.

Conclusion :

Pour le modèle final `modf` déterminé dans la partie 2 et pour le modèle complet, les résultats de prédiction sont à peu près les mêmes. Puisque le modèle final `modf` a une Residual Deviance plus petite que celle du modèle complet, le modèle final `modf` semble être un modèle pratique pour réaliser les prédictions. Le modèle final `modf` semble être un modèle pratique pour réaliser les prédictions sans avoir trop de faux négatifs mais possède un taux de faux positifs un peu trop conséquent.

Critiques :

- Certaines variables des tableaux de données ne sont pas significatives alors qu'elles devraient à priori jouer un rôle dans le diagnostic d'après des études cliniques (exemples : obésité, polyphagie...).
- Il ne semble pas y avoir de différences significatives entre le modèle final et le modèle complet en terme de prédiction malgré le fait que le modèle complet présente de nombreuses variables non significatives.
- Pour améliorer le modèle, on pourrait ajouter des variables quantitatives aux données comme par exemple la glycémie des patients.
- De plus, on ne sait pas de quel type de diabète il s'agit (1 ou 2). Le diabète de type 1 est une maladie immunitaire avec comme syndrome cardinal la polyurodipsie, l'amaigrissement et la polyphagie tandis que le diabète de type 2 est une maladie métabolique dans laquelle l'obésité est un facteur de risque très important. D'après les résultats, il semblerait donc qu'il s'agit ici d'une étude sur le diabète de type 1.