Projet MERR 2023

Colin Coërchon - Léos Coutrot

18-12-2023

Contents

Towards Parsimonious Model

Nous disposons d'un jeu de données avec 37 variables et 4424 observations obtenues grâce à des élèves de l'Institut polytechnique de Portalegre (Portugal) entre 2009 et 2019. Le jeu de données est complet et ne présente aucune donnée manquante. Elles sont, de plus, relativement riches (puisque nous avons n»p). Parmi les différentes variables à notre disposition, nous avons notamment le genre, l'âge, le niveau d'études de la mère ou du père (qui permettent d'avoir un aperçu de l'environnement familial des élèves).

La question centrale de notre travail est donc la suivante : **Quelles sont les variables ayant un impact significatif sur la performance scolaire des étudiants ?**

Pour répondre à cette interrogation, nous utiliserons la régression logistique, une méthode statistique robuste et adaptée pour modéliser les relations entre plusieurs variables et d'autant plus pertinente qu'ici notre variable cible est une variable catégorielle. Cette approche nous permettra de déterminer les facteurs les plus pertinents pour la réussite académique et d'établir un modèle prédictif pour identifier les étudiants à risque.

Les premières idées

Premièrement, la première idée qui nous vient naturellement serait de mettre de côté les variables non-significatives pour notre régression linéaire. Cette idée est très intéressantes, et nous avons ainsi vu dans le cours des méthodes de **sélections de variables**.

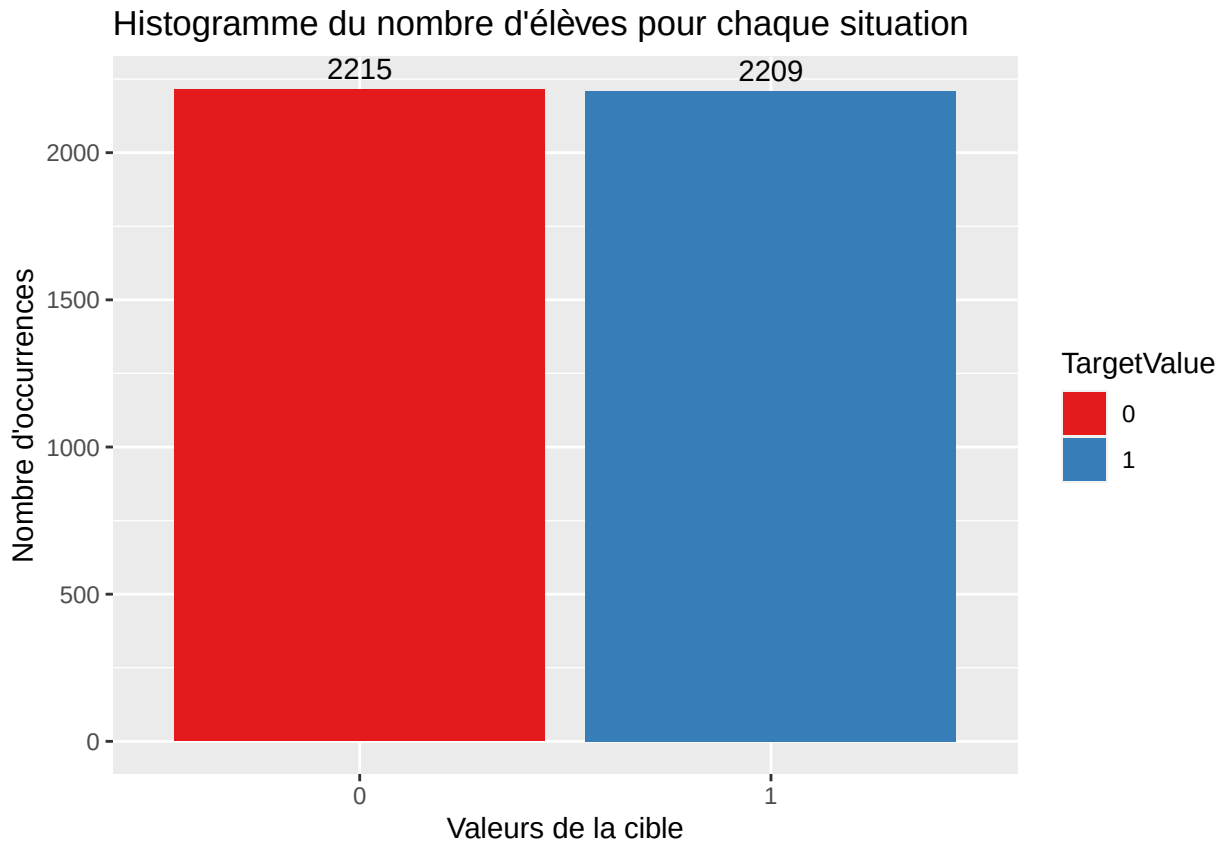
Pour tenter de nous approcher de ce sous-ensemble souhaité ($\widehat{\mathcal{M}}$), nous avons en effet découvert les algorithmes gloutons suivant en cours :

- "**Forward selection**" : On commence avec aucune variable et ajoute la variable la plus significative à chaque étape dans notre nouveau "Data Set" $\mathcal{M}$.
- "**Backward elimination**" : On commence avec toutes les variables et on élimine la moins significative à chaque étape.
- "**Stepwise regression**" : Une sorte d'entre d'eux des deux méthodes précédentes.

Ces méthodes sont dites incrémentales car elles ajoutent ou retirent une variable à la fois et réévaluent le modèle à chaque étape. Cependant, elles peuvent être assez gourmandes en temps de calcul et ne sont pas toujours pratiques lorsque p est grand (ce qui est un peu le cas ici ($p = 35$).

```r
# On charge le jeu de données dans R
data = read.csv("data.csv", header = TRUE, sep = ";")

# On converti les valeurs de la colonne cible pour qu'elle soit binaire
data$Target <- ifelse(data$Target == "Graduate", 1, 0)

# Compter le nombre d'occurrences pour chaque valeur dans la colonne 'Target'
counts <- table(data$Target)

# Convertir les comptages en dataframe
counts_df <- as.data.frame(counts)

# Renommer les colonnes pour plus de clarté
names(counts_df) <- c("TargetValue", "Frequency")

# Créer un histogramme avec ggplot, ajouter des étiquettes de texte et des couleurs
```

```r
ggplot(counts_df, aes(x = TargetValue, y = Frequency, fill = TargetValue)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = Frequency), vjust = -0.5) + # Ajouter les étiquettes de texte
  xlab("Valeurs de la cible") +
  ylab("Nombre d'occurrences") +
  ggtitle("Histogramme du nombre d'élèves pour chaque situation") +
  scale_fill_brewer(palette = "Set1") # Utiliser une palette de couleurs
```




```r
cols_to_convert <- c("Marital.status", "Application.mode", "Course", "Daytime.evening.attendance.",
                     "Previous.qualification", "Nacionality", "Mother.s.qualification",
                     "Father.s.qualification", "Mother.s.occupation", "Father.s.occupation",
                     "Displaced", "Educational.special.needs", "Debtor", "Tuition.fees.up.to.date",
                     "Gender", "Scholarship.holder", "International")

# On converti le type des variables catégorielles en "factor"
data[cols_to_convert] <- lapply(data[cols_to_convert], factor)
# On évalue l'information que porte la variable Educationnal special needs
table( "Target" = data$Target, "Special needs" = data$Educational.special.needs )
```

```
##        Special needs
## Target    0    1
##      0 2187   28
##      1 2186   23
```

```r
which(names(data) == "Educational.special.needs")
```

```
## [1] 15
```

```r
# On supprime la colonne "Educationnal.special.needs"
data <- subset(data, select = -15)

# On veut récupérer les catégories qui n'apparaissent qu'une seule fois
colonnes_categorielles <- sapply(data, is.factor)
nb_occurrences_par_colonne <- lapply(data[, colonnes_categorielles, drop = FALSE],
                                     function(col) {table(col)} )
categories_unicité_par_colonne <- lapply(nb_occurrences_par_colonne, function(t) {
  names(t[t == 1])
})
contient_categorie_unique <- function(row) {
  any(sapply(names(categories_unicité_par_colonne), function(col) {
    row[col] %in% categories_unicité_par_colonne[[col]]
  }))
}

# On supprime les observations ayant une catégorie unique
df_filtré <- data[!apply(data[, colonnes_categorielles, drop = FALSE], 1,
                         contient_categorie_unique), ]

df_filtré[, colonnes_categorielles] <- lapply(df_filtré[, colonnes_categorielles, drop = FALSE],
                                              function(col) {factor(col)} )

# Initialisation d'une liste pour stocker les données séparées par catégories
donnees_separees_list <- list()

# Boucle sur chaque nom de colonne catégorielle dans le dataframe filtré
for (col in names(df_filtré)[colonnes_categorielles]) {
    # Extraction des catégories uniques de la colonne en cours
    unique_categories <- unique(df_filtré[[col]])
    # Initialisation d'une liste pour stocker les sous-ensembles de données pour chaque catégorie
    category_list <- list()
    for (cat in unique_categories) {
        # Création d'un sous-ensemble de données où la colonne actuelle est égale à la catégorie en cou
        df_subset <- subset(df_filtré, df_filtré[[col]] == cat)
        if (nrow(df_subset) > 0) {
            # Ajout du premier enregistrement du sous-ensemble à la liste category_list
            category_list[[length(category_list) + 1]] <- df_subset[1, , drop = FALSE]
        }
    }
    donnees_separees_list[[col]] <- category_list
}


# Combinaison de tous les sous-ensembles de données en un seule dataframe
donnees_separees <- do.call(rbind, unlist(donnees_separees_list, recursive = FALSE))

# Création d'un dataframe df_restant en excluant les lignes déjà présentes dans donnees_separees et ide
df_restant <- df_filtré[!rownames(df_filtré) %in% rownames(donnees_separees), ]
```

```r
# Mélange aléatoire des lignes de df_restant sample(nrow(df_restant)) génère une séquence aléatoire de
df_restant <- df_restant[sample(nrow(df_restant)), ]

# Calcul de l'index pour diviser df_restant en données d'entraînement et de test, et arrondit à l'entie
index_split <- round(nrow(df_filtré) * 0.8)

# Création des données d'entraînement combine les données séparées et les premiers 80% de df_restant
train_data <- rbind(donnees_separees, df_restant[1:index_split, ])

# Création des données de test : combine les données séparées et les derniers 20% de df_restant
test_data <- rbind(donnees_separees, df_restant[(index_split + 1):nrow(df_restant), ])
```

Forward selection

```r
target <- train_data[, ncol(train_data)] # Cela extrait la dernière colonne comme variable cible
train_data <- train_data[, -ncol(train_data)] # Cela supprime la dernière colonne du jeu de données

base_model <- glm(Target ~ 1, data = data, family = "binomial")
```

```r
start_time <- proc.time()

forward_model <- step(base_model,
                      scope = list(lower = base_model,
                                   upper = glm(target ~ ., data = train_data, family = "binomial")),
                      direction = "forward", trace = FALSE)

end_time <- proc.time()

execution_time <- end_time - start_time
print(execution_time)
```

```
## utilisateur     système      écoulé
##       14.70        0.74       38.67
```

```r
# Extrait les informations du modèle
model_summary <- summary(forward_model)

# Crée un data frame avec les noms de variables, les coefficients, les erreurs standards,
# les z-values et les p-values
variables_info <- data.frame(
  Coefficient = model_summary$coefficients[, "Estimate"],
  StdError = model_summary$coefficients[, "Std. Error"],
  Zvalue = model_summary$coefficients[, "z value"],
  Pvalue = model_summary$coefficients[, "Pr(>|z|)"]
)


# Trie les variables par p-value en ordre croissant
variables_sorted <- variables_info[order(variables_info$Pvalue), ]
```

```r
# Affiche le résultat
head(variables_sorted, 10)
```

```
##                                              Coefficient   StdError    Zvalue
## Curricular.units.2nd.sem..approved.           0.85425884 0.06040760 14.141580
## Curricular.units.1st.sem..approved.           0.65355735 0.06396188 10.217919
## Tuition.fees.up.to.date1                      2.09794283 0.25784327  8.136504
## Scholarship.holder1                           0.70523011 0.11385453  6.194133
## Curricular.units.2nd.sem..enrolled.          -0.66636528 0.11174151 -5.963453
## Debtor1                                      -0.92338185 0.19001374 -4.859553
## Curricular.units.2nd.sem..evaluations.       -0.09731479 0.02716363 -3.582540
## (Intercept)                                  -6.31121676 1.88969757 -3.339803
## Curricular.units.2nd.sem..grade.              0.13420126 0.04522548  2.967381
## Curricular.units.2nd.sem..without.evaluations.  0.29926276 0.10567541  2.831905
##                                                    Pvalue
## Curricular.units.2nd.sem..approved.           2.105058e-45
## Curricular.units.1st.sem..approved.           1.648408e-24
## Tuition.fees.up.to.date1                      4.068550e-16
## Scholarship.holder1                           5.860668e-10
## Curricular.units.2nd.sem..enrolled.           2.469618e-09
## Debtor1                                       1.176511e-06
## Curricular.units.2nd.sem..evaluations.        3.402693e-04
## (Intercept)                                   8.383797e-04
## Curricular.units.2nd.sem..grade.              3.003482e-03
## Curricular.units.2nd.sem..without.evaluations. 4.627153e-03
```

Backward selection

Nous avons préféré commenter le code. Le temps de calcul étant vraiment trop long (entre 4 et 5 minutes en moyenne). **La méthode est bien fonctionnelle**, il faut juste avoir au moins 4 minutes devant soit pour le décommenter !

```r
# # Modèle de base avec toutes les variables
# base_model_backward <- glm(target ~ ., data = train_data, family = "binomial")
#
# # Backward selection
# start_time <- proc.time()
#
# backward_model <- step(base_model_backward, direction = "backward", trace = FALSE)
#
# end_time <- proc.time()
#
# execution_time <- end_time - start_time
# print(execution_time)
```

```r
# # Extrait les informations du modèle
# model_summary <- summary(backward_model)
#
# # Crée un data frame avec les noms de variables, les coefficients, les erreurs standards,
# # les z-values et les p-values
# variables_info <- data.frame(
#   Coefficient = model_summary$coefficients[, "Estimate"],
```

```
#   StdError = model_summary$coefficients[, "Std. Error"],
#   Zvalue = model_summary$coefficients[, "z value"],
#   Pvalue = model_summary$coefficients[, "Pr(>|z|)"]
# )
#
# variables_sorted <- variables_info[order(variables_info$Pvalue), ]
#
# head(variables_sorted, 10)
```

Stepwise selection

```
# Modèle de base pour la sélection stepwise
base_model_stepwise <- glm(target ~ 1, data = train_data, family = "binomial")

# Stepwise selection
start_time <- proc.time()

stepwise_model <- step(base_model_stepwise,
                       scope = list(lower = base_model_stepwise,
                                    upper = glm(target ~ ., data = train_data, family = "binomial")),
                       direction = "both", trace = FALSE)

end_time <- proc.time()

execution_time <- end_time - start_time
print(execution_time)
```

```
## utilisateur     système      écoulé
##       20.23        0.73       50.58
```

```
# Extrait les informations du modèle
model_summary <- summary(stepwise_model)

# Crée un data frame avec les noms de variables, les coefficients, les erreurs standards,
# les z-values et les p-values
variables_info <- data.frame(
  Coefficient = model_summary$coefficients[, "Estimate"],
  StdError = model_summary$coefficients[, "Std. Error"],
  Zvalue = model_summary$coefficients[, "z value"],
  Pvalue = model_summary$coefficients[, "Pr(>|z|)"]
)


# Trie les variables par p-value en ordre croissant
variables_sorted <- variables_info[order(variables_info$Pvalue), ]

# Affiche le résultat
head(variables_sorted, 10)
```

```
##                                          Coefficient   StdError    Zvalue
```

```
## Curricular.units.2nd.sem..approved.              0.89618552 0.06626503 13.524260
## Curricular.units.1st.sem..approved.              0.64662668 0.06297715 10.267639
## Curricular.units.2nd.sem..enrolled.             -0.87257396 0.10232664 -8.527339
## Tuition.fees.up.to.date1                         1.92454785 0.26558123  7.246551
## Curricular.units.1st.sem..credited.             -0.26109159 0.05667350 -4.606943
## Scholarship.holder1                              0.56691582 0.12919655  4.388011
## Curricular.units.2nd.sem..without.evaluations.   0.47650409 0.11138009  4.278180
## Debtor1                                         -0.86636032 0.21453441 -4.038328
## Father.s.qualification2                         -1.23394136 0.39167375 -3.150432
## Curricular.units.2nd.sem..evaluations.          -0.08501009 0.02991627 -2.841600
##                                                       Pvalue
## Curricular.units.2nd.sem..approved.             1.124702e-41
## Curricular.units.1st.sem..approved.             9.858773e-25
## Curricular.units.2nd.sem..enrolled.             1.497522e-17
## Tuition.fees.up.to.date1                        4.275188e-13
## Curricular.units.1st.sem..credited.             4.086311e-06
## Scholarship.holder1                             1.143922e-05
## Curricular.units.2nd.sem..without.evaluations. 1.884274e-05
## Debtor1                                         5.383354e-05
## Father.s.qualification2                         1.630294e-03
## Curricular.units.2nd.sem..evaluations.          4.488774e-03
```

Matrice de confusion

```r
# Prédiction avec forward_model
Y_pred_forward <- predict(forward_model, newdata = test_data, type = "response")

# Affichage des valeurs prédites minimales et maximales
cat("min: ", min(Y_pred_forward), "\nmax: ", max(Y_pred_forward), "\n")
```

```
## min:  2.292792e-06
## max:  0.9978486
```

```r
# Seuil pour définir les classes
Y_pred_class_forward <- ifelse(Y_pred_forward > 0.5, 1, 0)

# Matrice de confusion
confusion_matrix_forward <- table("True class" = test_data$Target,
                                  "Prediction" = Y_pred_class_forward)
print(confusion_matrix_forward)
```

```
##           Prediction
## True class   0   1
##          0 444  98
##          1  48 478
```

```r
precision<- sum(diag(confusion_matrix_forward)) / sum(confusion_matrix_forward)
cat("precision: ", precision)
```

```
## precision:  0.8632959
```

```r
# # Prédiction avec backward_model
# Y_pred_backward <- predict(backward_model, newdata = test_data, type = "response")
#
# # Affichage des valeurs prédites minimales et maximales
# cat("min: ", min(Y_pred_backward), "\nmax: ", max(Y_pred_backward), "\n")
#
# # Seuil pour définir les classes
# Y_pred_class_backward <- ifelse(Y_pred_backward > 0.5, 1, 0)
#
# # Matrice de confusion
# confusion_matrix_backward <- table("True class" = test_data$Target,
#                                    "Prediction" = Y_pred_class_backward)
# print(confusion_matrix_backward)
#
# precision<- sum(diag(confusion_matrix_backward)) / sum(confusion_matrix_backward)
# cat("precision: ", precision)
```

```r
# Prédiction avec stepwise_model
Y_pred_stepwise <- predict(stepwise_model, newdata = test_data, type = "response")

# Affichage des valeurs prédites minimales et maximales
cat("min: ", min(Y_pred_stepwise), "\nmax: ", max(Y_pred_stepwise), "\n")
```

```
## min:  5.684126e-12
## max:  0.9999999
```

```r
# Seuil pour définir les classes
Y_pred_class_stepwise <- ifelse(Y_pred_stepwise > 0.5, 1, 0)

# Matrice de confusion
confusion_matrix_stepwise <- table("True class" = test_data$Target,
                                   "Prediction" = Y_pred_class_stepwise)
print(confusion_matrix_stepwise)
```

```
##           Prediction
## True class   0   1
##          0 445  97
##          1  42 484
```

```r
precision<- sum(diag(confusion_matrix_stepwise)) / sum(confusion_matrix_stepwise)
cat("precision: ", precision)
```

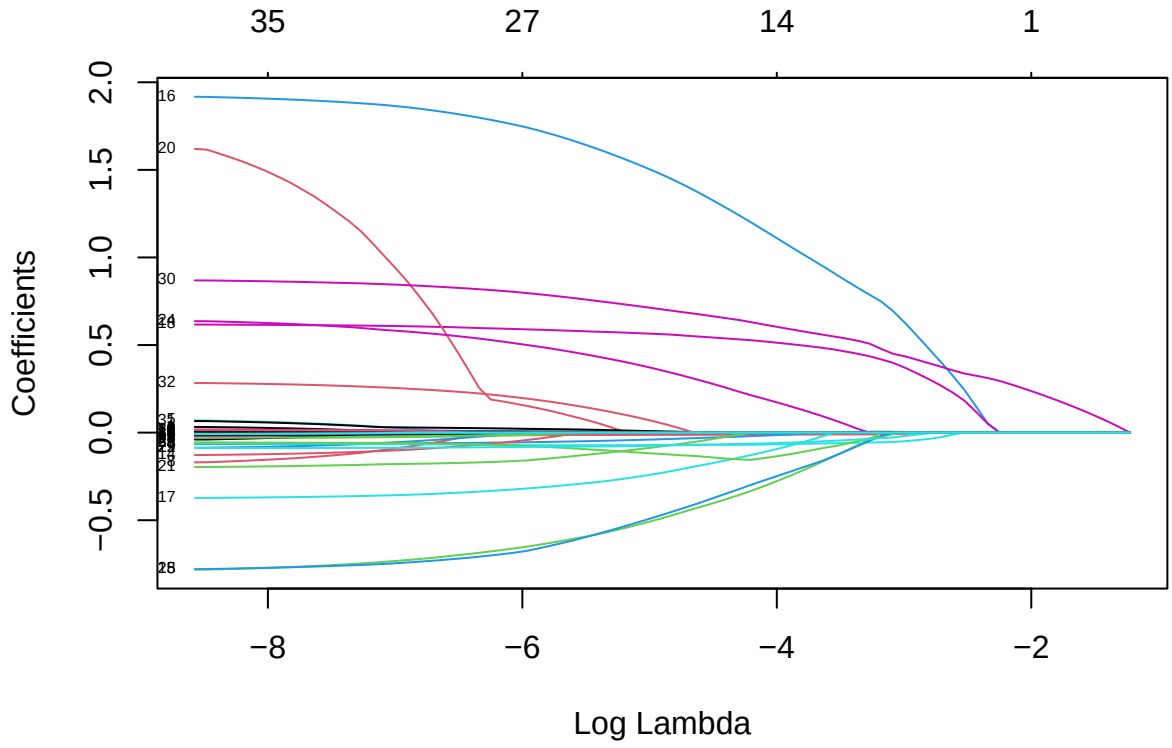
```
## precision:  0.8698502
```

Modèles intermédiaires

Pénalisations Ridge et Lasso

Lasso

```
lasso_model <- glmnet(train_data, target, alpha = 1, family = "binomial")
plot(lasso_model,xvar = "lambda", label = TRUE)
```




```
# On calcule les lambda optimaux (via une cross validation) et et les modèles avec
# les bons lambdas correspondants
cv_lasso <- cv.glmnet(as.matrix(train_data), target, alpha = 1, family = "binomial",
                      grouped=FALSE)
lasso_min <- glmnet(as.matrix(train_data), target, alpha = 1, family = "binomial",
                    lambda = cv_lasso$lambda.min)
lasso_1se <- glmnet(as.matrix(train_data), target, alpha = 1, family = "binomial",
                    lambda = cv_lasso$lambda.1se)

print(cv_lasso$lambda.min==cv_lasso$lambda.1se)
```

```
## [1] FALSE
```

Nous allons évaluer :

```r
# Fonction pour tester le modèle et générer la matrice de confusion
test_model <- function(model, test_data) {
  Y_pred <- predict(model, newx = as.matrix(test_data[,-which(names(test_data)=="Target")]),
                    type = "response")
  # Seuil MAP
  Y_pred_class <- ifelse(Y_pred > 0.5, 1, 0)
  return(table("True class"=test_data$Target,"Prediction"=Y_pred_class))
}

# Test modèle Lasso min
# Modèle pour lambda min
test_model(lasso_min,test_data)
```

```
##           Prediction
## True class   0   1
##          0 440 102
##          1  45 481
```

```r
conf_mat<-test_model(lasso_min,test_data = test_data)
accuracy_min<- sum(diag(conf_mat)) / sum(conf_mat)
cat("Précision : ",accuracy_min,"\n")
```

```
## Précision :  0.8623596
```

```r
#Modèle pour lambda 1se
test_model(lasso_1se,test_data)
```

```
##           Prediction
## True class   0   1
##          0 437 105
##          1  47 479
```

```r
conf_mat<-test_model(lasso_1se,test_data = test_data)
accuracy_min<- sum(diag(conf_mat)) / sum(conf_mat)
cat("Précision : ",accuracy_min,"\n")
```

```
## Précision :  0.8576779
```

```r
#Récupérer les variable importantes
variable_names <- c("Intercept", colnames(train_data))
coef_lasso <- coef(lasso_min, s = cv_lasso$lambda.min)
names_lasso <- variable_names[which(coef_lasso != 0, arr.ind = TRUE)[, 1]]
lasso_results <- data.frame(Variable = names_lasso, Coefficient = coef_lasso[coef_lasso != 0])
lasso_results <- lasso_results[order(abs(lasso_results$Coefficient), decreasing = TRUE), ]
head(lasso_results, 10)
```

```
##                                        Variable Coefficient
## 1                                     Intercept  -2.9515578
## 14                    Tuition.fees.up.to.date   1.8437643
## 27 Curricular.units.2nd.sem..approved.   0.8168839
```

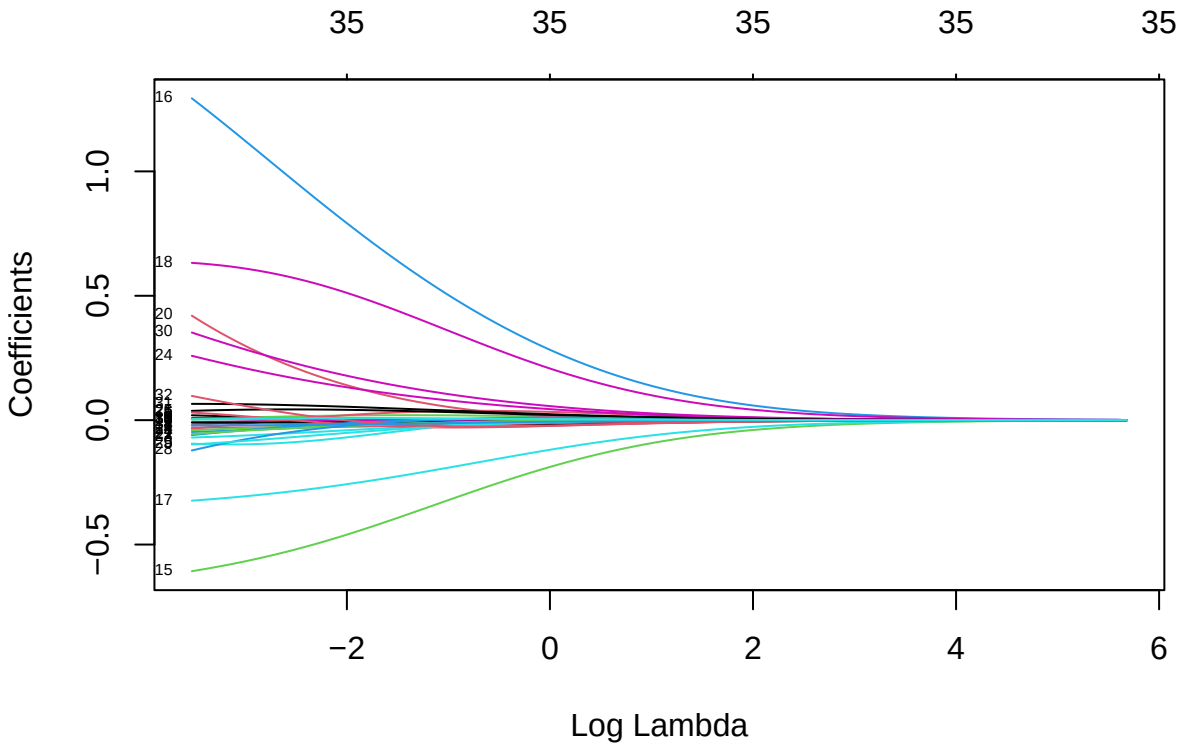
```
## 25            Curricular.units.2nd.sem..enrolled.  -0.7339329
## 18                                 International   0.7309993
## 13                                       Debtor  -0.7181641
## 16                           Scholarship.holder   0.6057590
## 22            Curricular.units.1st.sem..approved.   0.5582079
## 15                                       Gender  -0.3368335
## 29 Curricular.units.2nd.sem..without.evaluations.   0.2208452
```

```r
#Récupérer les variable importantes
variable_names <- c("Intercept", colnames(train_data))
coef_lasso <- coef(lasso_1se, s = cv_lasso$lambda.1se)
names_lasso <- variable_names[which(coef_lasso != 0, arr.ind = TRUE)[, 1]]
lasso_results <- data.frame(Variable = names_lasso, Coefficient = coef_lasso[coef_lasso != 0])
# On trie nos coefficients par ordre décroissant
lasso_results <- lasso_results[order(abs(lasso_results$Coefficient), decreasing = TRUE), ]
head(lasso_results, 10)
```

```
##                                Variable Coefficient
## 1                             Intercept -2.98200488
## 7                 Tuition.fees.up.to.date  1.47523202
## 16    Curricular.units.2nd.sem..approved.  0.70255311
## 9                     Scholarship.holder  0.56451111
## 14    Curricular.units.2nd.sem..enrolled. -0.48803119
## 6                                 Debtor -0.47285940
## 12    Curricular.units.1st.sem..approved.  0.34689085
## 8                                 Gender -0.21178215
## 13    Curricular.units.2nd.sem..credited. -0.11335742
## 15 Curricular.units.2nd.sem..evaluations. -0.06848433
```

Ridge

```r
ridge_model <- glmnet(train_data, target, alpha = 0, family = "binomial")
plot(ridge_model,xvar = "lambda", label = TRUE)
```




```
# On calcule les lambda optimaux (via une cross validation) et et les modèles avec
# les bons lambdas correspondants
cv_ridge <- cv.glmnet(as.matrix(train_data), target, alpha = 0, family = "binomial",
                      grouped=FALSE)
ridge_min <- glmnet(as.matrix(train_data), target, alpha = 0, family = "binomial",
                    lambda = cv_ridge$lambda.min)
ridge_1se <- glmnet(as.matrix(train_data), target, alpha = 0, family = "binomial",
                    lambda = cv_ridge$lambda.1se)

print(cv_ridge$lambda.min==cv_ridge$lambda.1se)
```

```
## [1] FALSE
```

Nous allons évaluer

```
# On test les modèles

# Modèle pour lambda min
test_model(ridge_min,test_data = test_data)
```

```
##            Prediction
## True class   0   1
##           0 435 107
##           1  53 473
```

```r
conf_mat<-test_model(ridge_min,test_data = test_data)
accuracy_min<- sum(diag(conf_mat)) / sum(conf_mat)
cat("Précision : ",accuracy_min,"\n")
```

```
## Précision :  0.8501873
```

```r
#Modèle pour lambda 1se
test_model(ridge_1se,test_data = test_data)
```

```
##           Prediction
## True class   0   1
##          0 435 107
##          1  54 472
```

```r
conf_mat<-test_model(ridge_1se,test_data = test_data)
accuracy_min<- sum(diag(conf_mat)) / sum(conf_mat)
cat("Précision : ",accuracy_min,"\n")
```

```
## Précision :  0.8492509
```

```r
#Récupérer les variable importantes
variable_names <- c("Intercept", colnames(train_data))
coef_ridge <- coef(ridge_min, s = cv_ridge$lambda.min)
names_ridge <- variable_names[which(coef_ridge != 0, arr.ind = TRUE)[, 1]]
ridge_results <- data.frame(Variable = names_ridge, Coefficient = coef_ridge[coef_ridge != 0])
# On trie les coefficients par ordre décroissant
ridge_results <- ridge_results[order(abs(ridge_results$Coefficient), decreasing = TRUE), ]
head(ridge_results, 10)
```

```
##                                        Variable Coefficient
## 1                                     Intercept -3.17945941
## 17                Tuition.fees.up.to.date  1.30351932
## 19                     Scholarship.holder  0.62535716
## 16                                   Debtor -0.61288683
## 21                            International  0.50763048
## 31    Curricular.units.2nd.sem..approved.  0.34500547
## 18                                   Gender -0.31902121
## 25    Curricular.units.1st.sem..approved.  0.25408215
## 29    Curricular.units.2nd.sem..enrolled. -0.10440442
## 33 Curricular.units.2nd.sem..without.evaluations.  0.09475815
```

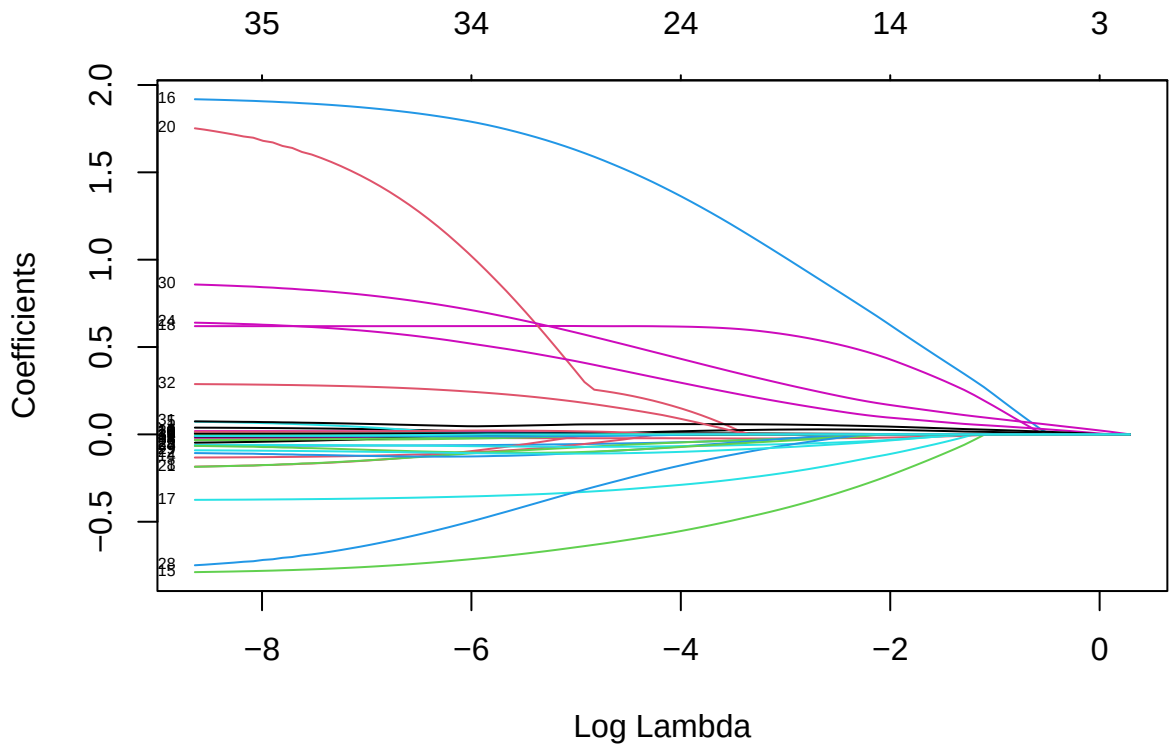
```r
#Récupérer les variable importantes
variable_names <- c("Intercept", colnames(train_data))
coef_ridge <- coef(ridge_1se, s = cv_ridge$lambda.1se)
names_ridge <- variable_names[which(coef_ridge != 0, arr.ind = TRUE)[, 1]]
ridge_results <- data.frame(Variable = names_ridge, Coefficient = coef_ridge[coef_ridge != 0])
# On trie les coefficients par ordre décroissant
ridge_results <- ridge_results[order(abs(ridge_results$Coefficient), decreasing = TRUE), ]
head(ridge_results, 10)
```

```
##                                  Variable Coefficient
## 1                                Intercept -3.17878936
## 17               Tuition.fees.up.to.date  1.21217639
## 19                    Scholarship.holder  0.61535586
## 16                                Debtor -0.59123055
## 21                           International  0.43478625
## 18                                Gender -0.31055230
## 31    Curricular.units.2nd.sem..approved.  0.30979027
## 25    Curricular.units.1st.sem..approved.  0.22771781
## 30 Curricular.units.2nd.sem..evaluations. -0.08725942
## 6             Daytime.evening.attendance. -0.08194696
```

Modèle avancé

Elastic net

```
elastic_model <- glmnet(train_data, target, alpha = 0.22, family = "binomial")
plot(elastic_model,xvar = "lambda", label = TRUE)
```




```
alpha_grid <- seq(0, 1, by=0.05)
cv_results <- lapply(alpha_grid, function(a) {
    cv.glmnet(as.matrix(train_data), target,type.measure ="class" ,alpha=a,family="binomial")
})
cv_errors <- sapply(cv_results, function(cv) min(cv$cvm))
best_alpha <- alpha_grid[which.min(cv_errors)]

# On calcule les lambda optimaux (via une cross validation) et et les modèles avec
# les bons lambdas correspondants
cv_elastic <- cv.glmnet(as.matrix(train_data), target, alpha = best_alpha, family = "binomial",
                        grouped=FALSE)
elastic_min <- glmnet(as.matrix(train_data), target, alpha = best_alpha, family = "binomial",
                      lambda = cv_elastic$lambda.min)
elastic_1se <- glmnet(as.matrix(train_data), target, alpha = best_alpha, family = "binomial",
                      lambda = cv_elastic$lambda.1se)

print(cv_elastic$lambda.min==cv_elastic$lambda.1se)
```

```
## [1] FALSE
```

Nous allons évaluer

```
# On test les modèles
# Modèle pour lambda min
test_model(elastic_min,test_data = test_data)
```

```
##           Prediction
## True class   0   1
##          0 440 102
##          1  45 481
```

```
conf_mat<-test_model(elastic_min,test_data = test_data)
accuracy_min<- sum(diag(conf_mat)) / sum(conf_mat)
cat("Précision : ",accuracy_min,"\n")
```

```
## Précision :  0.8623596
```

```
#Modèle pour lambda 1se
test_model(elastic_1se,test_data = test_data)
```

```
##           Prediction
## True class   0   1
##          0 440 102
##          1  49 477
```

```
conf_mat<-test_model(elastic_1se,test_data = test_data)
accuracy_min<- sum(diag(conf_mat)) / sum(conf_mat)
cat("Précision : ",accuracy_min,"\n")
```

```
## Précision :  0.8586142
```

```
#Récupérer les variable importantes
variable_names <- c("Intercept", colnames(train_data))
coef_elastic <- coef(elastic_1se, s = cv_elastic$lambda.1se)
names_elastic <- variable_names[which(coef_elastic != 0, arr.ind = TRUE)[, 1]]
elastic_results <- data.frame(Variable = names_elastic,
                              Coefficient = coef_elastic[coef_elastic != 0])
# On trie les coefficients par ordre décroissant
elastic_results <- elastic_results[order(abs(elastic_results$Coefficient), decreasing = TRUE), ]
head(elastic_results, 10)
```

```
##                                       Variable Coefficient
## 1                                    Intercept  -3.2587542
## 12                     Tuition.fees.up.to.date   1.5432221
## 14                          Scholarship.holder   0.6002092
## 26          Curricular.units.2nd.sem..approved.   0.5731003
## 11                                    Debtor  -0.5650059
## 20          Curricular.units.1st.sem..approved.   0.3782154
```

```
## 24            Curricular.units.2nd.sem..enrolled.  -0.3251872
## 13                                        Gender  -0.2781699
## 28 Curricular.units.2nd.sem..without.evaluations.   0.1208253
## 23            Curricular.units.2nd.sem..credited.  -0.1087164
```

```r
#Récupérer les variable importantes
variable_names <- c("Intercept", colnames(train_data))
coef_elastic <- coef(elastic_min, s = cv_elastic$lambda.min)
names_elastic <- variable_names[which(coef_elastic != 0, arr.ind = TRUE)[, 1]]
elastic_results <- data.frame(Variable = names_elastic,
                              Coefficient = coef_elastic[coef_elastic != 0])
# On trie les coefficients par ordre décroissant
elastic_results <- elastic_results[order(abs(elastic_results$Coefficient), decreasing = TRUE), ]
head(elastic_results, 10)
```

```
##                                         Variable Coefficient
## 1                                      Intercept  -2.8773718
## 17                      Tuition.fees.up.to.date   1.8785925
## 21                                International   0.9844308
## 30           Curricular.units.2nd.sem..approved.   0.7939364
## 16                                      Debtor  -0.7559191
## 28           Curricular.units.2nd.sem..enrolled.  -0.6706315
## 19                           Scholarship.holder   0.6193891
## 25           Curricular.units.1st.sem..approved.   0.5821364
## 18                                        Gender  -0.3527193
## 32 Curricular.units.2nd.sem..without.evaluations.   0.2481798
```

Annexe

ACP

```
library(FactoMineR)
```

```
## Warning: le package 'FactoMineR' a été compilé avec la version R 4.3.2
```

```
library(factoextra)
```

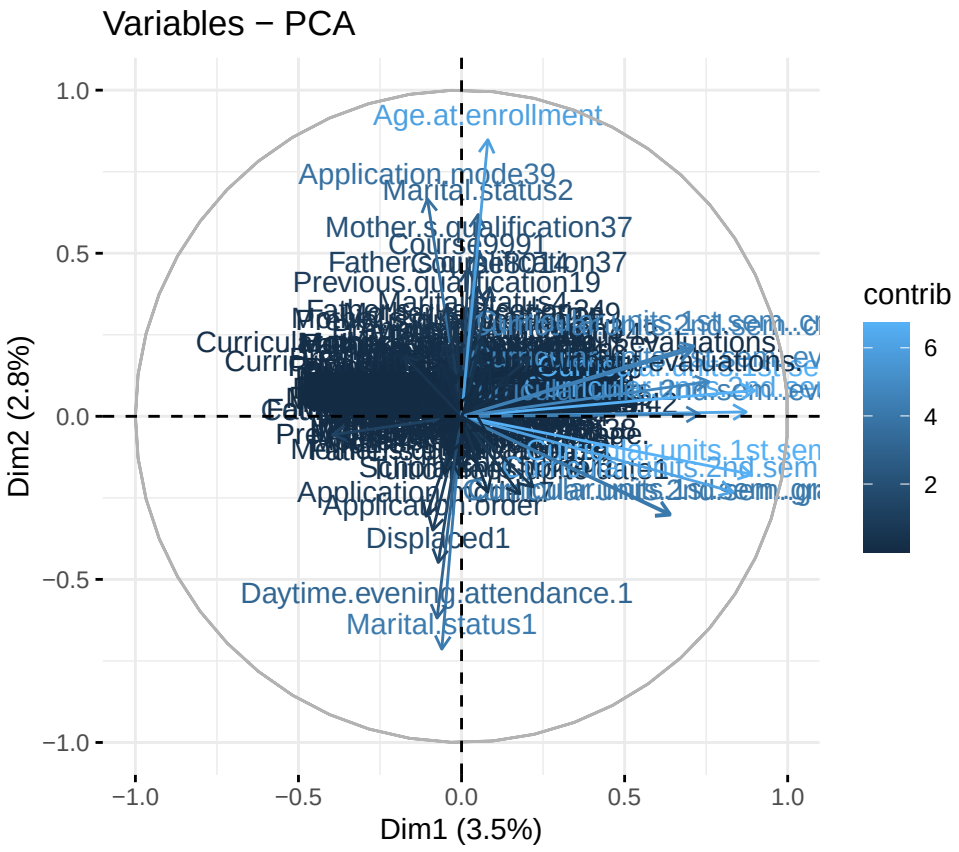
```
## Warning: le package 'factoextra' a été compilé avec la version R 4.3.2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

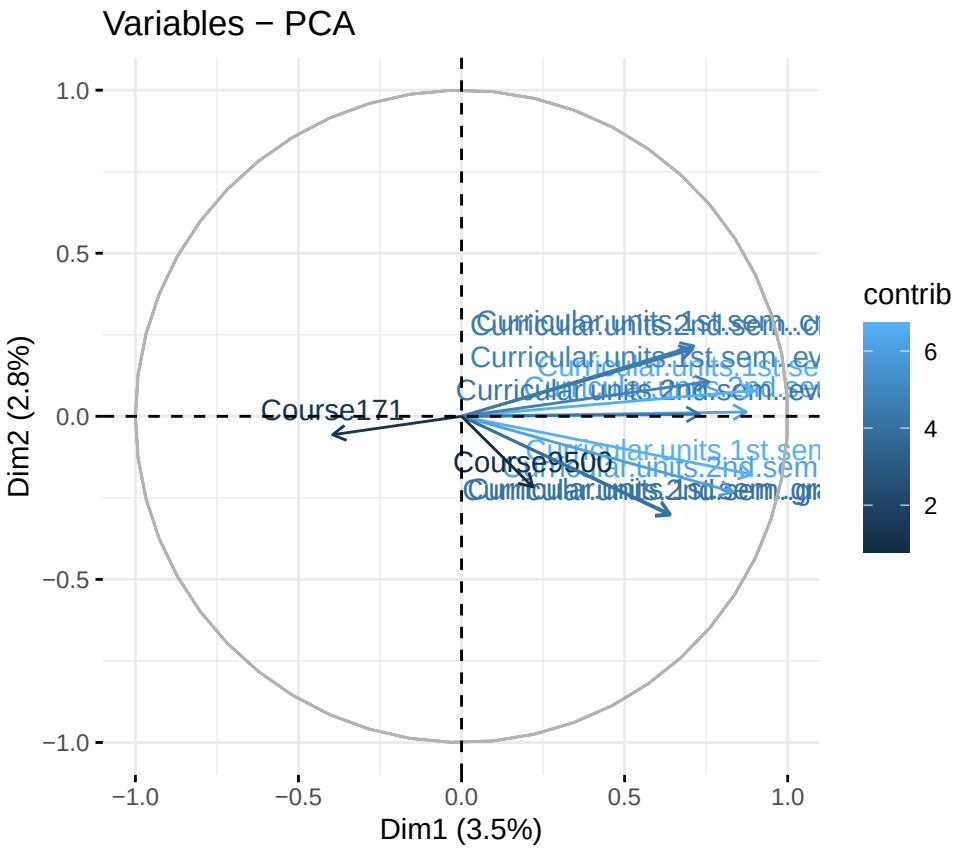
```
library(dplyr)
library(tibble)
```

```
## Warning: le package 'tibble' a été compilé avec la version R 4.3.2
```

```
# Convertir les facteurs en variables indicatrices (dummy variables)
train_data_dummy <- model.matrix(~ . - 1, data = train_data)

# Effectuer l'ACP sur les données avec variables indicatrices
res.pca <- PCA(train_data_dummy, graph = FALSE)

# Visualiser le cercle de corrélation
fviz_pca_var(res.pca, col.var = "contrib")
```




```r
# Convertir les facteurs en variables indicatrices (dummy variables)
train_data_dummy <- model.matrix(~ . - 1, data = train_data)

# Effectuer l'ACP
res.pca <- PCA(train_data_dummy, graph = FALSE)

# Calculer les contributions et trier
contributions <- as.data.frame(res.pca$var$contrib)
top_contributors <- contributions %>%
  rownames_to_column("variable") %>%
  arrange(desc(Dim.1)) %>%
  head(12) %>%
  pull(variable)

# Visualiser le cercle de corrélation seulement pour les variables les plus contributives
fviz_pca_var(res.pca, col.var = "contrib",
             select.var = list(name = top_contributors))
```


Variables − PCA
1.0
0.5
0.0
−0.5
−1.0
Dim2 (2.8%)
−1.0
−0.5
0.0
0.5
1.0
Dim1 (3.5%)
Course171
Course9500
contrib
6
4
2