Compte Rendu TP

2022-01-08

Importation des librairies

```
library(lares)
library(caret)
library(ggplot2)
library(aod)
library(GGally)
library(broom.helpers)
library(questionr)
library(effects)
library(DescTools)
library(tree)
library(randomForest)
library(rpart)
library(rpart.plot)
```

Importation des données

```
students <- read.table("students.csv", sep = ";", header = TRUE)
```

Première approche des données

On regarde à quoi ressemble les premières lignes du dataset et on constate qu'il y a 4 variables sur lesquelles on va tenter d'en apprendre plus

```
head(students)
```

```
##   admit gre  gpa rank
## 1     0 380 3.61    3
## 2     1 660 3.67    3
## 3     1 800 4.00    1
## 4     1 640 3.19    4
## 5     0 520 2.93    4
## 6     1 760 3.00    2
```

On constate que le dataset contient les données de 400 individus et les variables sont les suivantes :

- `admit` variable permettant de voir si l'individu est admis (1) ou non (0)
- `gre` variable représentant possiblement le résultat à un test passé par les individus du dataset
- `gpa` variable représentant possiblement un indice de performance des résultats scolaires des individus
- `rank` variable pouvant représenter le rang de l'établissement scolaire de l'individu

```
str(students)
```

```
## 'data.frame':    400 obs. of  4 variables:
```

```
## $ admit: int 0 1 1 1 0 1 1 0 1 0 ...
## $ gre  : int 380 660 800 640 520 760 560 400 540 700 ...
## $ gpa  : num 3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
## $ rank : int 3 3 1 4 4 2 1 2 3 2 ...
```

Avec l'analyse descriptive, on constate que les variables `rank`et `admit` sont considérées comme des variables numériques, ce qui est incorrect car elles doivent être traitées comme des variables catégoriques

```
summary(students)
```

```
##      admit             gre             gpa             rank      
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000  
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000  
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485  
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000  
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
```

```
students$admit <- as.factor(students$admit)
students$rank <- as.factor(students$rank)
summary(students)
```

```
##  admit        gre             gpa         rank   
##  0:273   Min.   :220.0   Min.   :2.260   1: 61  
##  1:127   1st Qu.:520.0   1st Qu.:3.130   2:151  
##          Median :580.0   Median :3.395   3:121  
##          Mean   :587.7   Mean   :3.390   4: 67  
##          3rd Qu.:660.0   3rd Qu.:3.670          
##          Max.   :800.0   Max.   :4.000
```

Ici on vérifie que les données ne comportent pas de valeurs manquantes pour chaque colonne (ce qui est bien le cas)

```
sapply(students, function(x) sum(is.na(x)))
```

```
## admit   gre   gpa  rank 
##     0     0     0     0
```

On vérifie que les groupes d'individus pour la variable cible sont relativement bien équilibrés. Ici les effectifs reste relativement équilibrés mais il est possible que notre modèle soit affecté par la surreprésentation de 0 pour la variable `admit`

```
table(students$admit)
```

```
## 
##   0   1 
## 273 127
```

On split les données en deux datasets afin de pouvoir tester la précision du modèle plus tard

```
set.seed(125)
ind <- createDataPartition(students$admit, p = 0.80, list = FALSE)
students <- students[ind, ] #train
students_test <- students[-ind, ] #test
str(students)
```

```
## 'data.frame':    321 obs. of  4 variables:
##  $ admit: Factor w/ 2 levels "0","1": 1 2 2 1 2 2 1 1 1 2 ...
##  $ gre  : int  380 660 640 520 760 540 700 800 440 760 ...
##  $ gpa  : num  3.61 3.67 3.19 2.93 3 3.39 3.92 4 3.22 4 ...
```

```
##  $ rank : Factor w/ 4 levels "1","2","3","4": 3 3 4 4 2 3 2 4 1 1 ...
```

```
str(students_test)
```

```
## 'data.frame':    67 obs. of  4 variables:
##  $ admit: Factor w/ 2 levels "0","1": 2 1 1 2 2 1 1 1 2 1 ...
##  $ gre  : int  640 700 800 580 740 560 620 580 620 500 ...
##  $ gpa  : num  3.19 3.92 4 3.46 4 3.32 3.3 4 4 2.71 ...
##  $ rank : Factor w/ 4 levels "1","2","3","4": 4 2 4 2 3 4 1 2 1 2 ...
```

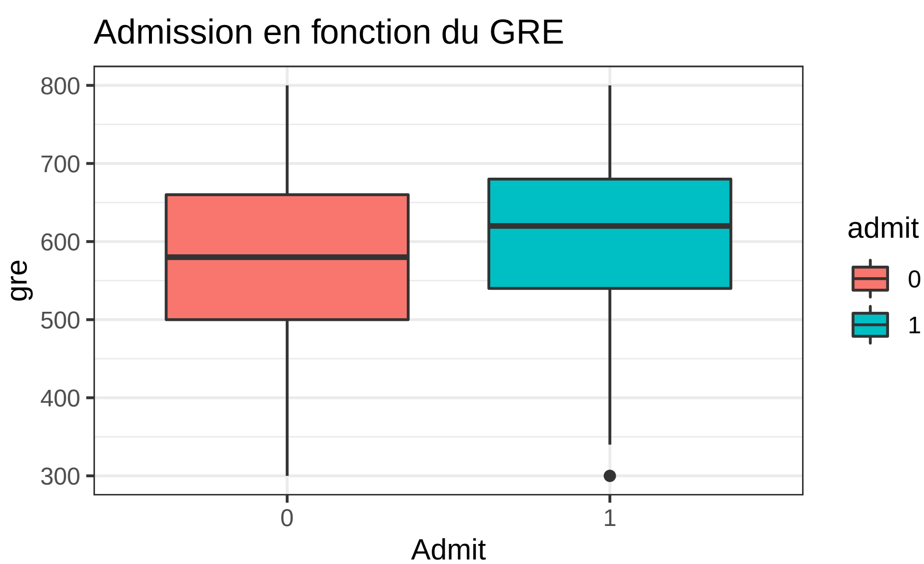
Analyse descriptive

Dans un premier temps on peut étudier le lien entre les variables à l'aide boxplot (pour les variables quantitatives) et barplot (pour les categories).

gre

On constate que les personnes admises ont obtenu un `gre` plus élevé que celles non admises. Il est donc possible à première vue que cette variable soit significative. Néanmoins cet écart n'est pas très élévé et la taille de l'effectif peut aussi impacter cette analyse.

```
ggplot(students, aes(admit, gre, fill = admit)) +
  geom_boxplot() +
  theme_bw() +
  xlab("Admit") +
  ylab("gre") +
  ggtitle("Admission en fonction du GRE")
```

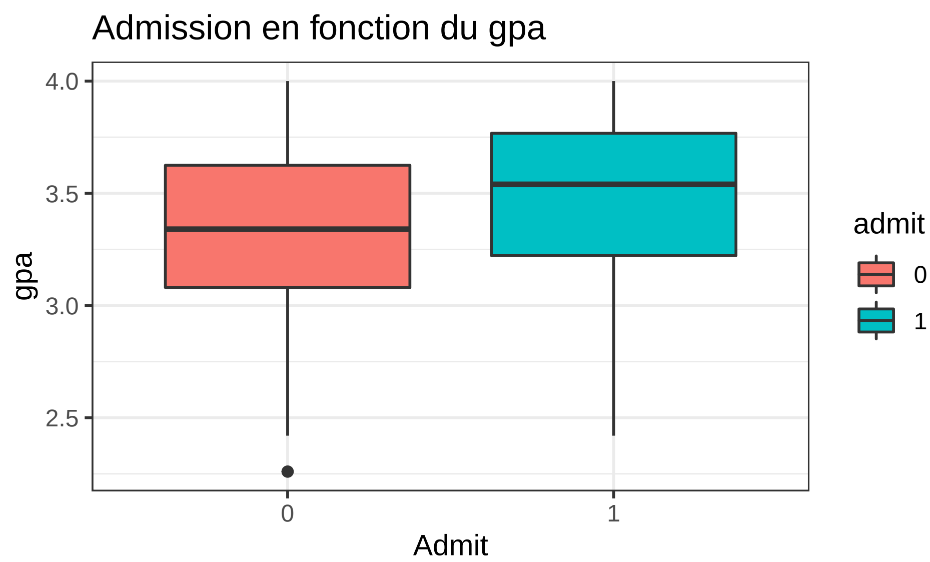


gpa

Similairement au `gre`, on constate que les personnes admises ont obtenu un `gpa` plus élevé que celles non admises. Il est donc possible à première vue que cette variable soit significative. L'écart entre les deux boxplot est un peu plus grand que pour la variable `gre` mais il demeure assez petit.

```
ggplot(students, aes(admit, gpa, fill = admit)) +
  geom_boxplot() +
  theme_bw() +
```

```
  xlab("Admit") +
  ylab("gpa") +
  ggtitle("Admission en fonction du gpa")
```

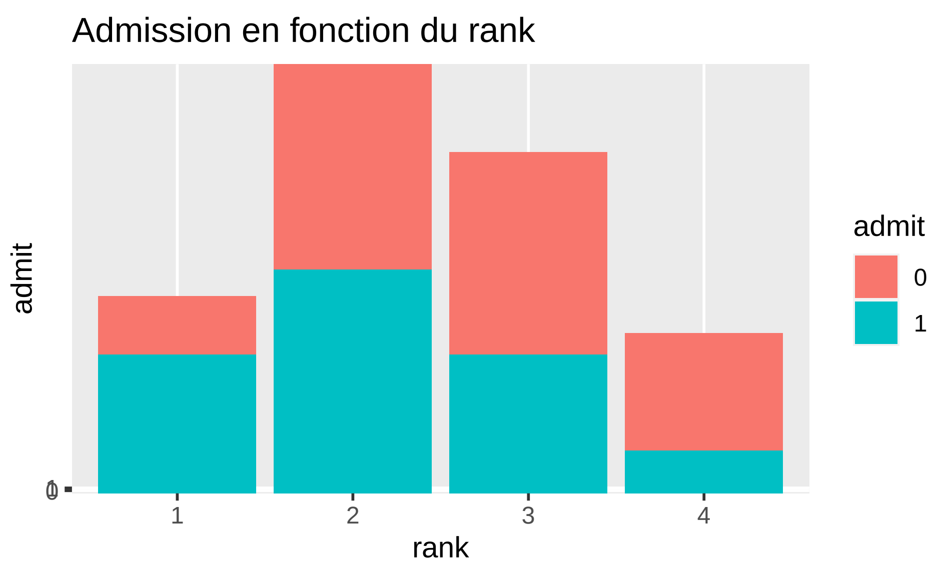


rank

Comme `rank` est une variable catégorique, il est plus approprié d'utiliser un barplot.

Ici on observe qu'il existe une tendance qui montre que plus le `rank` est proche de 1, plus les chances d'admissions sont grandes.

```
ggplot(students, aes(rank, admit,fill = admit)) +
  geom_col() +
  xlab("rank") +
  ylab("admit") +
  ggtitle("Admission en fonction du rank")
```



Afin de savoir si réellement les variables `gre` `gpa` et `rank` ont un impact sur `admit`, on va mettre en place plusieurs modèles qui de manière générale permettent d'identifier des relations entre variables

Regression Logistique

Les lignes suivantes sont effectuées dans le cadre du modèle linéaire généralisé (GLM)

On effectue la regression logistique en laissant tous les predicteurs

```
glm_fm1 <- glm(admit ~., data = students, family = "binomial")
coef(glm_fm1)
```

```
##  (Intercept)          gre          gpa        rank2        rank3        rank4
## -3.433909961  0.001798479  0.722890231 -0.694880868 -1.197652452 -1.757036892
```

A l'issue de la regression, on constate que toutes les variables sont statistiquement significatives sauf `gre` (p-value = 0.136 > 0.05)

De plus pour la variable `rank`, la valeur 1 est la valeur de reférence pour la regression (par exemple selon le modèle, la probabilité d'être admis en provenant d'un établissement de rang 3 est exp(-1.197652) = 0.302 fois celle d'un établissement de rang 1)

```
summary(glm_fm1)
```

```
##
## Call:
## glm(formula = admit ~ ., family = "binomial", data = students)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5641  -0.8879  -0.6505   1.1669   2.1205
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.433910   1.222502  -2.809 0.004971 **
## gre          0.001798   0.001207   1.489 0.136370
## gpa          0.722890   0.360032   2.008 0.044659 *
## rank2       -0.694881   0.355228  -1.956 0.050447 .
## rank3       -1.197652   0.376338  -3.182 0.001461 **
## rank4       -1.757037   0.488519  -3.597 0.000322 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 401.36  on 320  degrees of freedom
## Residual deviance: 370.77  on 315  degrees of freedom
## AIC: 382.77
##
## Number of Fisher Scoring iterations: 4
```

En effectuant une selection backward et forward, on constate que toutes les variables sont gardées par l'algorithme donc `gre` semble bien avoir un impact sur `admit`.

```
back_sel <- step(glm_fm1, direction = "backward")
```

```
## Start:  AIC=382.77
## admit ~ gre + gpa + rank
```

```
##
##        Df Deviance    AIC
## <none>      370.77 382.77
## - gre   1   373.02 383.02
## - gpa   1   374.89 384.89
## - rank  3   388.47 394.47
```

```r
summary(back_sel)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = "binomial",
##     data = students)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5641  -0.8879  -0.6505   1.1669   2.1205
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.433910   1.222502  -2.809 0.004971 **
## gre          0.001798   0.001207   1.489 0.136370
## gpa          0.722890   0.360032   2.008 0.044659 *
## rank2       -0.694881   0.355228  -1.956 0.050447 .
## rank3       -1.197652   0.376338  -3.182 0.001461 **
## rank4       -1.757037   0.488519  -3.597 0.000322 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 401.36  on 320  degrees of freedom
## Residual deviance: 370.77  on 315  degrees of freedom
## AIC: 382.77
##
## Number of Fisher Scoring iterations: 4
```

```r
glm_fm2 <- glm(admit ~ 1, data = students, family = "binomial")
back_sel2 <- step(glm_fm2, direction = "forward", scope = list(lower = glm_fm2, upper = ~ gpa+gre+rank))
```

```
## Start:  AIC=403.36
## admit ~ 1
##
##        Df Deviance    AIC
## + rank  3   381.18 389.18
## + gpa   1   391.71 395.71
## + gre   1   393.11 397.11
## <none>      401.36 403.36
##
## Step:  AIC=389.18
## admit ~ rank
##
##        Df Deviance    AIC
## + gpa   1   373.02 383.02
## + gre   1   374.89 384.89
```

```
## <none>      381.18 389.18
##
## Step:  AIC=383.02
## admit ~ rank + gpa
##
##        Df Deviance    AIC
## + gre   1   370.77 382.77
## <none>      373.02 383.02
##
## Step:  AIC=382.77
## admit ~ rank + gpa + gre
```

```
summary(back_sel2)
```

```
##
## Call:
## glm(formula = admit ~ rank + gpa + gre, family = "binomial",
##     data = students)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5641  -0.8879  -0.6505   1.1669   2.1205
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.433910   1.222502  -2.809 0.004971 **
## rank2       -0.694881   0.355228  -1.956 0.050447 .
## rank3       -1.197652   0.376338  -3.182 0.001461 **
## rank4       -1.757037   0.488519  -3.597 0.000322 ***
## gpa          0.722890   0.360032   2.008 0.044659 *
## gre          0.001798   0.001207   1.489 0.136370
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 401.36  on 320  degrees of freedom
## Residual deviance: 370.77  on 315  degrees of freedom
## AIC: 382.77
##
## Number of Fisher Scoring iterations: 4
```

Intervalle de confiance des coefficients

On peut representer les coefficients (+ intervalle de confiance) obtenus après la regression et après exponentiation.

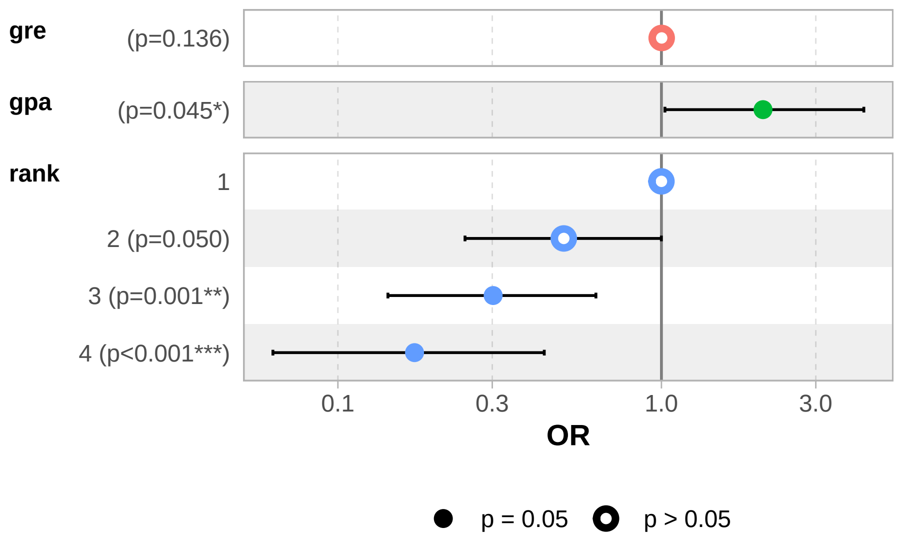
Ces coefficients montrent que le `gpa` augmente la probabilité d'admission (car > 1). A l'inverse un `rank` égal à 3 ou 4, diminue la probabilité par rapport à un `rank` égal à 1. Le `gre` ne semble pas impacter le modèle mais les différentes selections ci-dessus le conserve.

```
odds.ratio(glm_fm1) # ou exp(cbind(coef(glm_fm1), confint(glm_fm1)))
```

```
##                    OR     2.5 % 97.5 %         p
## (Intercept) 0.0322606 0.0027873 0.3409 0.0049708 **
## gre         1.0018001 0.9994473 1.0042 0.1363702
```

```
## gpa         2.0603796 1.0250803 4.2224 0.0446591 *
## rank2       0.4991339 0.2470971 0.9995 0.0504467 .
## rank3       0.3019021 0.1427707 0.6273 0.0014607 **
## rank4       0.1725554 0.0629803 0.4343 0.0003223 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ggcoef_model(glm_fm1, exponentiate = TRUE)
```



Effet de la variable rank

Pour vérifier si la variable `rank` est vraiment significative, on peut mettre en place un test de Wald. Ce test permet de tester la nullité ou non du predicteur `rank`

On definit :

- H0: la valeur du paramètre `rank` dans le modèle est nulle
- H1: la valeur du paramètre `rank` dans le modèle n'est pas nulle

On obtient ensuite une p-value largement inférieure à 0.005, donc on peut rejeter H0 et la variable `rank` est significative

```
wald.test(b = coef(glm_fm1), Sigma = vcov(glm_fm1), Terms = 4:6)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 16.6, df = 3, P(> X2) = 0.00087
```

Precision du modèle

On utilise le dataset de test que l'on a construit tout à l'heure, afin de tester l'accuracy de notre modèle.

On obtient une accuracy de 74,6% et une specificité de 95% ce qui est correct malgré la faible sensitivité (32%). Ce modèle a donc tendance a bien prédire les vrais négatifs (ceux non admis) mais il a des difficultés à bien trouver les vrais positifs.

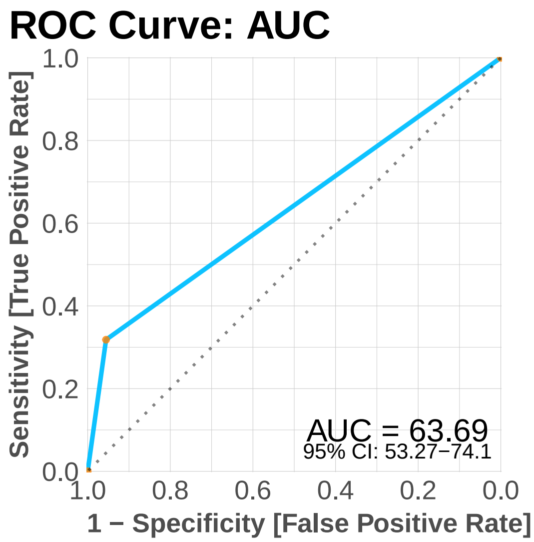
```r
prediction <- predict(glm_fm1, students_test, type = "response")
prediction <- ifelse(prediction >= 0.5, 1, 0)
prediction <- as.factor(prediction)
confusionMatrix(prediction, students_test$admit, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 43 15
##          1  2  7
##
##                Accuracy : 0.7463
##                  95% CI : (0.6251, 0.8447)
##     No Information Rate : 0.6716
##     P-Value [Acc > NIR] : 0.119523
##
##                   Kappa : 0.3224
##
##  Mcnemar's Test P-Value : 0.003609
##
##             Sensitivity : 0.3182
##             Specificity : 0.9556
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7414
##              Prevalence : 0.3284
##          Detection Rate : 0.1045
##    Detection Prevalence : 0.1343
##       Balanced Accuracy : 0.6369
##
##        'Positive' Class : 1
##
```

Pouvoir discriminant du modèle

A partir de la courbe ROC, on calcule l'AUC. On obtient 64% ce qui peut paraitre un peu faible mais cela s'explique par la taille de l'échantillon (assez faible) et par le desequilibre des effectifs pour la variable `admit`

```r
tag <- as.numeric(students_test$admit)
score <- as.numeric(prediction)
mplot_roc(tag = tag, score = score)
```



Ajustement du modèle

Comme pour les regressions linéaires, il est possible de tester la qualité d'ajustement du modèle grâce à des ratios.

Ici, on utilise le pseudo R^2 de McFadden qui permet de mesurer la qualité de l'ajustement (grace aux valeurs du log likelihood du modèle nul et du modèle ajusté pseudo R^2 = 1-LLmod/LL0). Un bon modèle possède un pseudo R2 de McFadden compris en 0,2 et 0,4. Ici on obtient une valeur plus faible ce qui montre que le modèle n'est pas forcement très ajuste (LLMod est presque à LL0 donc le modèle ajusté n'est pas vraiment meilleur que le modèle nul sur ce point)

```
PseudoR2(glm_fm1, which = "McFadden")
```

```
##   McFadden
## 0.07620342
```

Significativité du modèle

On peut ensuite passer à l'étude de la significativité du modèle afin de voir si le modèle avec les predicteurs apporte plus d'informations que le modèle nul.

Pour cela on met en place un test du chi-2 avec :

- H0: les deux modèles (nul et avec predicteurs) décrivent aussi bien le modèle
- H1: le modèle avec predicteurs colle plus aux données

On obtient ensuite une p-value largement inférieure à 0.005, donc on peut rejeter H0 et effectivement le modèle avec predicteurs est statistiquement significatif pour décrire la relation entre les données.

```
with(glm_fm1, null.deviance - deviance) #Difference de deviance = chi-2
```

```
## [1] 30.5848
```

```
with(glm_fm1, df.null - df.residual) #nb de DDL
```

```
## [1] 5
```

```
with(glm_fm1, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE)) #p-value
```

```
## [1] 1.131196e-05
```

On peut aussi regarder les différents graphiques (QQ-Plot, Residual VS Fitted, etc. . .) mais sous GLM il est plus difficile d'intepreter ces graphiques que pour une regression linéaire.

Conclusion

D'après le modèle que l'on a construit, toutes les variables ont un impact sur l'admission des élèves. Pour le `rank`, plus il est proche de 1, plus il est facile d'être admis. Le `gpa` possède aussi un impact important ce qui semble là aussi logique dans la mesure où cette variable rend compte des notes d'un élève durant son cursus. Cependant comme mentionné précédemment, la taille de l'échantillon et le fait que pseudo R2 et l'AUC soit assez faible amène à se poser des questions sur la pertinence du modèle

Arbre de décision

On va utiliser un modèle produisant un arbre de décision car les arbres peuvent être utiles pour des problèmes de classification.

```
tree <- rpart(admit ~., data = students)
```

On constate que ici, la variable `gpa` semble être celle qui est la plus importante pour definir les chances d'admissions d'un étudiant, ensuite les variables `gre` et `rank` possederaient la même importance.

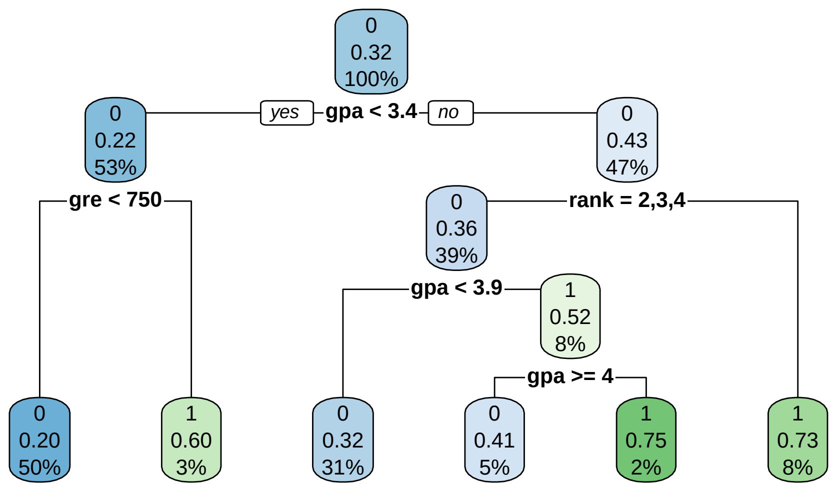
```
summary(tree)
```

```
## Call:
## rpart(formula = admit ~ ., data = students)
##   n= 321
##
##           CP nsplit rel error   xerror       xstd
## 1 0.05882353      0 1.0000000 1.000000 0.08178421
## 2 0.01960784      2 0.8823529 1.009804 0.08199628
## 3 0.01000000      5 0.8235294 1.186275 0.08512458
##
## Variable importance
##  gpa rank  gre
##   46   29   24
##
## Node number 1: 321 observations,    complexity param=0.05882353
##   predicted class=0  expected loss=0.317757  P(node) =1
##     class counts:   219   102
##    probabilities: 0.682 0.318
##   left son=2 (171 obs) right son=3 (150 obs)
##   Primary splits:
##       gpa  < 3.445 to the left,  improve=6.679792, (0 missing)
##       rank splits as  RLLL,      improve=5.659255, (0 missing)
##       gre  < 510   to the left,  improve=4.204299, (0 missing)
##   Surrogate splits:
##       gre  < 630   to the left,  agree=0.657, adj=0.267, (0 split)
##       rank splits as  RLLL,      agree=0.545, adj=0.027, (0 split)
##
## Node number 2: 171 observations,    complexity param=0.01960784
##   predicted class=0  expected loss=0.2222222  P(node) =0.5327103
##     class counts:   133    38
##    probabilities: 0.778 0.222
##   left son=4 (161 obs) right son=5 (10 obs)
##   Primary splits:
##       gre  < 750   to the left,  improve=3.0316080, (0 missing)
```

```
##       rank splits as  RRLL,      improve=1.9446270, (0 missing)
##       gpa  < 2.685 to the right, improve=0.5551511, (0 missing)
##
## Node number 3: 150 observations,    complexity param=0.05882353
##   predicted class=0  expected loss=0.4266667  P(node) =0.4672897
##     class counts:    86    64
##    probabilities: 0.573 0.427
##   left son=6 (124 obs) right son=7 (26 obs)
##   Primary splits:
##       rank splits as  RLLL,      improve=5.8171880, (0 missing)
##       gre  < 550   to the right, improve=0.9685380, (0 missing)
##       gpa  < 3.945 to the left,  improve=0.9282488, (0 missing)
##
## Node number 4: 161 observations
##   predicted class=0  expected loss=0.1987578  P(node) =0.5015576
##     class counts:   129    32
##    probabilities: 0.801 0.199
##
## Node number 5: 10 observations
##   predicted class=1  expected loss=0.4  P(node) =0.03115265
##     class counts:     4     6
##    probabilities: 0.400 0.600
##
## Node number 6: 124 observations,    complexity param=0.01960784
##   predicted class=0  expected loss=0.3629032  P(node) =0.3862928
##     class counts:    79    45
##    probabilities: 0.637 0.363
##   left son=12 (99 obs) right son=13 (25 obs)
##   Primary splits:
##       gpa  < 3.945 to the left,  improve=1.5455780, (0 missing)
##       rank splits as  -RLL,      improve=0.9347766, (0 missing)
##       gre  < 690   to the right, improve=0.4354839, (0 missing)
##
## Node number 7: 26 observations
##   predicted class=1  expected loss=0.2692308  P(node) =0.08099688
##     class counts:     7    19
##    probabilities: 0.269 0.731
##
## Node number 12: 99 observations
##   predicted class=0  expected loss=0.3232323  P(node) =0.3084112
##     class counts:    67    32
##    probabilities: 0.677 0.323
##
## Node number 13: 25 observations,    complexity param=0.01960784
##   predicted class=1  expected loss=0.48  P(node) =0.07788162
##     class counts:    12    13
##    probabilities: 0.480 0.520
##   left son=26 (17 obs) right son=27 (8 obs)
##   Primary splits:
##       gpa  < 3.995 to the right, improve=1.2447060, (0 missing)
##       rank splits as  -RLL,      improve=1.0800000, (0 missing)
##       gre  < 730   to the left,  improve=0.7339683, (0 missing)
##   Surrogate splits:
##       gre < 690   to the right, agree=0.72, adj=0.125, (0 split)
```

```
##
## Node number 26: 17 observations
##    predicted class=0  expected loss=0.4117647  P(node) =0.0529595
##      class counts:    10     7
##     probabilities: 0.588 0.412
##
## Node number 27: 8 observations
##    predicted class=1  expected loss=0.25  P(node) =0.02492212
##      class counts:     2     6
##     probabilities: 0.250 0.750
```

```
rpart.plot(tree)
```



Precision du modèle

Pour évaluer la qualité du modèle, on peut s'attarder sur la matrice de confusion pour l'échantillon de test. On retrouve des valeurs similaires à celle de la regression logistique, ce qui montre que le modèle est plutôt pertinent même si il reste fortement perfectible.

```
p <- predict(tree, students_test, type = "class")
confusionMatrix(p, students_test$admit, positive = "1")
```

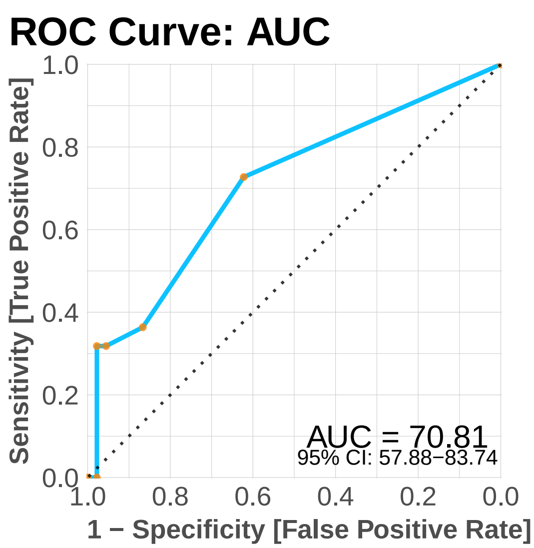
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 43 15
##          1  2  7
##
##                Accuracy : 0.7463
##                  95% CI : (0.6251, 0.8447)
##     No Information Rate : 0.6716
##     P-Value [Acc > NIR] : 0.119523
##
##                   Kappa : 0.3224
##
##  Mcnemar's Test P-Value : 0.003609
```

```
##
##             Sensitivity : 0.3182
##             Specificity : 0.9556
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7414
##              Prevalence : 0.3284
##          Detection Rate : 0.1045
##    Detection Prevalence : 0.1343
##       Balanced Accuracy : 0.6369
##
##        'Positive' Class : 1
##
```

Pouvoir discriminant du modèle

Pour l'AUC que l'on obtient à partir de la courbe ROC, on obtient une valeur supérieure à celle de la regression logistique.

```
p1 <- predict(tree, students_test, type="prob")
p1 <- p1[,2]
tag <- as.numeric(students_test$admit)
p1 <- as.numeric(p1)
mplot_roc(tag = tag, score = p1)
```



Conclusion

D'après ce modèle, le `gpa` est ce qui joue le plus dans les probabilités d'être admis. Ce qui confirme la tendance de la regression logistique. Les autres variables ont aussi une importance moindre. Néanmoins, ce modèle semble connaitre quelque limite comme pour la regression logistique.

Random Forest

Pour finir, on va essayer de construire un modèle grâce à une technique d'apprentissage nommée Random Forest, cette technique est très efficace pour les problèmes de classification.

```
rf_pima <- randomForest(admit ~., data = students, proximity=TRUE)
print(rf_pima)
```

```
##
## Call:
##  randomForest(formula = admit ~ ., data = students, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 28.04%
## Confusion matrix:
##     0  1 class.error
## 0 205 14  0.06392694
## 1  76 26  0.74509804
```

Precision du modèle

Ici, on constate que de manière générale cette méthode de classification semble plus performante et donc on peut possiblement en tirer plus de conclusions. En effet, la précision du modèle est nettement supérieure à celle des deux précédents (80%), de même la specificité est parfaite et la sensitivité est en augmentation même si elle reste assez faible.

```r
rf_probs <- predict(rf_pima, students_test)
# bonne accuracy et specificité
confusionMatrix(rf_probs, students_test$admit,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 45 13
##          1  0  9
##
##                Accuracy : 0.806
##                  95% CI : (0.6911, 0.8924)
##     No Information Rate : 0.6716
##     P-Value [Acc > NIR] : 0.0110310
##
##                   Kappa : 0.4819
##
##  Mcnemar's Test P-Value : 0.0008741
##
##             Sensitivity : 0.4091
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.7759
##              Prevalence : 0.3284
##          Detection Rate : 0.1343
##    Detection Prevalence : 0.1343
##       Balanced Accuracy : 0.7045
##
##        'Positive' Class : 1
##
```

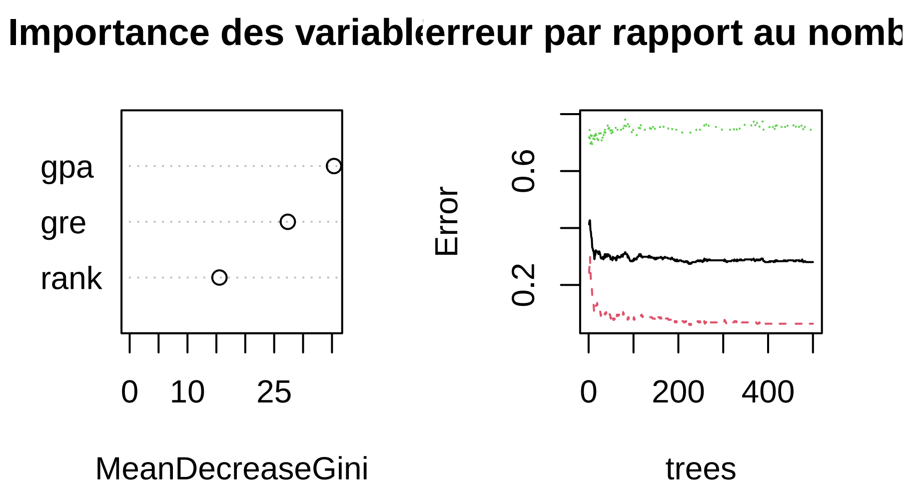
Importance des variables

On constate que pour ce modèle, le `gpa` est la variable la plus importante puis suivent le `gre` et le `rank`. Ce qui confirme les tendances des modèles précédents.

```
importance(rf_pima)
```

```
##      MeanDecreaseGini
## gre          27.35536
## gpa          35.34857
## rank         15.53023
```

```
par(mfrow = c(1, 2))
varImpPlot(rf_pima, type = 2, main = "Importance des variables",col = 'black')
plot(rf_pima, main = "Taux d'erreur par rapport au nombre d'arbres")
```



Conclusion

Les différents modèles affirment tous (pour l'échantillon donné) que les résultats au lycée ont un impact sur les probabilités d'être admis dans le supérieur. Chaque modèle possède des specificités sur l'importance des variables mais il semble de la `gpa` soit un bon indicateur suivi du `gpe` et du `rank`. Néanmoins comme évoqué précédemment, il faudrait avoir un échantillon plus important pour tirer des conclusions plus certaines.