Lecture 10 : Methods for Regression

Logistic model/Classification: study case

K. Meziani

Dauphine | PSL
UNIVERSITÉ PARIS

Section 1

1. Logistic model under R

Packages

```
library(MASS); library(knitr);library(ggplot2)
library(cowplot);library(reshape2);library(dplyr)
library(GGally);library(corrplot);library(carData)
library(car);library(questionr);library(multcomp)
library(dplyr);library(tidyverse);library(forestmodel)
library(effects);library(pscl);library(ResourceSelection)
library(survey);library(caret);library(pROC);library(ROCR)
library(mlr);library(randomForest);library(party)
library(rpart);library(rpart.plot);library(effects)
library(nnet);library(ResourceSelection)
```

The "binary.csv" dataset

We are interested here in the admission of students to a new establishment:

- The response variable ("**admit**") is a factor with 2 modalities ("0" = not allowed and "1" = allowed).
- To explain the admission, we have 2 covariates: "**gre**" and "\texttt{**gpa**"
- and the rank of the establishment of origin: the factor "**rank**" which admits 4 modalities ("1", "2", "3" and "4"). Note that a rank "1" establishment is more famous than a rank 2 establishment ...

Upload the dataset

```
mydata=read.csv("binary.csv",header=T,sep=",")
names(mydata)
```

```
## [1] "admit" "gre"   "gpa"   "rank"
```

Nature of the data

```
str(mydata)
```

```
## '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 ...
```

```
mydata$admit=factor(mydata$admit)
mydata$rank=factor(mydata$rank)
```

Importance of the reference modality

- The order of the modalities of the response variable matters.
- The reference modality **must** correspond to the fact of not fulfilling the criterion studied:
  - here not to be admitted (“0”). **Note from now on that in the library caret, the reference class is considered as the "*positive*" class.**

```
levels(mydata$admit)
```

```
## [1] "0" "1"
```

Summary

```
summary(mydata)
```

```
##  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
```

The plan is complet and unbalanced !

Split the dataset (*train* and *test*)

```
set.seed(1234)
ind=sample(2,nrow(mydata),replace=T,prob=c(0.8,0.2))
train=mydata[ind==1,]
test=mydata[ind==2,]
dim(train)
```

```
## [1] 325   4
```

```
dim(test)
```

```
## [1] 75  4
```

Modelisation

On se place dans le cadre de la régression logistique où la variable réponse est binaire, *i.e.* de type facteur admettant 2 modalités : “0/1” ou “Non/Oui”, etc. . . :

- “$Y_i = 1$” (ou “Y_i = Oui”) si l’individu i possède la caractéristique étudiée.
- “$Y_i = 0$” (ou “Y_i = Non”) sinon.
- $X^{(1)}$=‘**gre**’, $X^{(2)}$=‘**gpa**’, F=‘**rank**’

$$x_{ij}^T \beta = \mu + \alpha_j + \beta_1 X_{ij}^{(1)} + \beta_2 X_{ij}^{(2)}$$

no interaction

- Y_{ij}=‘**admitt**’ $\in \{0, 1\} \rightarrow$ choose $Y_{ij}|x_{ij} \sim \mathcal{B}(p(x_{ij}))$

$$\mathbb{E}[Y_{ij}|x_{ij}] = p(x_{ij}) \quad \text{and} \quad p(x_{ij}) = \frac{e^{x_{ij}^T \beta}}{1 + e^{x_{ij}^T \beta}}.$$

Declaration of the model under R (2 ways)

```
#library(nnet)
Mmod=multinom(admit~.,data=train)
```

```
## # weights:  7 (6 variable)
## initial  value 225.272834
## iter  10 value 184.997478
## final  value 184.997466
## converged
```

les deux donnent le bien logit.

```
mod=glm(admit~.,family='binomial',data=train)
summary(mod)
```

Here constraint by default : $\alpha_1 = \alpha_{rank1} = 0$.

Declaration of the model under R (2 ways)

```
##
## Call:
## glm(formula = admit ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5873  -0.8679  -0.6181   1.1301   2.1178
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.009514   1.316514  -3.805 0.000142 ***
## gre          0.001631   0.001217   1.340 0.180180
## gpa          1.166408   0.388899   2.999 0.002706 **
## rank2       -0.570976   0.358273  -1.594 0.111005
## rank3       -1.125341   0.383372  -2.935 0.003331 **
## rank4       -1.532942   0.477377  -3.211 0.001322 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 404.39  on 324  degrees of freedom
## Residual deviance: 369.99  on 319  degrees of freedom
## AIC: 381.99
##
## Number of Fisher Scoring iterations: 4
```

Discussion

```
Deviance Residuals:
Min       1Q   Median       3Q      Max
-1.5873  -0.8679  -0.6181   1.1301   2.1178
```

see previous Lecture.

```
AIC: 381.99
```

AIC of our model

```
Number of Fisher Scoring iterations : 4
```

pour la convergence de l'algo de NR.

Needed iterations to converge. Fisher's scoring algorithm is a derivative of Newton's method for solving maximum likelihood problems numerically.

Discussion

```
Residual deviance: 369.99  on 319  degrees of freedom
```

corresponds to the value of the deviance of our model [mod]

$$\mathcal{D}_{[\text{mod}]}(\widehat{p}) = 2(\mathcal{L}_{[m_{sat}]}(\widehat{p}) - \mathcal{L}_{[\text{mod}]}(\widehat{p})) = -2\mathcal{L}_{[\text{mod}]}(\widehat{p})$$

```
Null deviance: 404.39  on 324  degrees of freedom
```

corresponds to the value of the deviance of the null model (reduced to the intercept).

$$\mathcal{D}_{[\text{nul}]}(\widehat{p}) = -2\mathcal{L}_{[\text{nul}]}(\widehat{p}) = -2\mathcal{L}_{[\text{nul}]}(\overline{Y})$$

Si la déviance du modèle nul est meilleure, on préfère le modèle réduit à l'intercept.

Discussion

	z value	Pr(>|z|)	
(Intercept)	-3.805	0.000142	***
gre	1.340	0.180180	
gpa	2.999	0.002706	**
rank2	-1.594	0.111005	
rank3	-2.935	0.003331	**
rank4	-3.211	0.001322	**

Line by line, we test the nullity of the coefficients by a student test.

Wald tests with library(survey)

```
regTermTest(mod,"gre")
```

```
## Wald test for gre
##  in glm(formula = admit ~ ., family = "binomial", data = train)
## F =  1.796141  on  1  and  319  df: p= 0.18113
```

```
regTermTest(mod,"gpa")
```

```
## Wald test for gpa
##  in glm(formula = admit ~ ., family = "binomial", data = train)
## F =  8.995527  on  1  and  319  df: p= 0.0029194
```

```
regTermTest(mod,"rank")
```

```
## Wald test for rank
##  in glm(formula = admit ~ ., family = "binomial", data = train)
## F =  4.653994  on  3  and  319  df: p= 0.0033647
```

c'est du Fischer

Here the variable `gre` does not seem significant.

Plot

- Puisque la réponse n'est pas supposée suivre une distribution normale, nous ne nous intéressons pas vraiment au diagramme quantile-quantile.
- Le graphique Residuals vs Fitted permet de vérifier l'absence de tendance dans les résidus. Pour chaque valeur prédite (ici, les prédictions sont représentées sur l'échelle du prédicteur linéaire, pas celle de p. Il n'y a que deux valeurs possibles pour le résidu: l'une positive si la réponse est 1, et l'autre négative si la réponse est 0. C'est pourquoi on observe deux lignes de points sur le graphique ci-dessus.
- Le graphique Residuals vs Leverage permet de détecter des points avec une grande influence sur la régression.

Plot

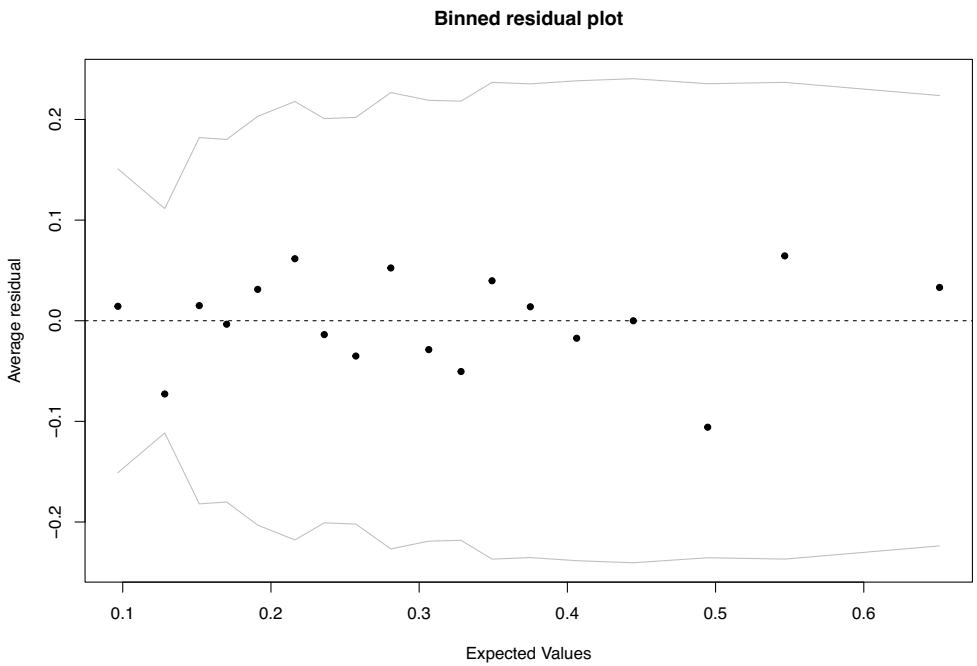
Notez que trois des graphiques utilisent les résidus de Pearson (Std. Pearson resid.) qui correspondent aux résidus divisés par l'écart-type attendu de la réponse (rappelons-nous que cet écart-type dépend de la valeur attendue de la réponse ici). Ces résidus devraient donc avoir une variance plus homogène que les résidus bruts.

Plot

Alternative à Residuals vs Fitted: Une meilleure stratégie serait de regrouper les points avec des probabilités prédites semblables, puis calculer le résidu correspondant à la différence entre (1) la proportion de réponses positives observées parmi les points d'un groupe et (2) la probabilité moyenne pour ces points. Par exemple, pour un groupe de 20 points avec 11 réponses positives et une prédiction moyenne de 0.6, le résidu serait -0.05 (11/20 - 0.6). La fonction binnedplot du package arm sert à créer un tel graphique des résidus groupés (binned residual plot).

Plot

```
library(arm)
binnedplot(fitted(mod), residuals(mod, type = "response"))
```



Plot

Note: Il existe différentes définitions des résidus pour un modèle linéaire généralisé, donc il faut spécifier un type dans la fonction residuals. Pour ce graphique-ci, nous utilisons les résidus sur l'échelle de la réponse (type = "response"), soit les différences entre les réponses observées et prédites (

$$y - \widehat{y}$$

Par défaut, binnedplot choisit le nombre de groupes d'après un compromis visant à avoir suffisamment de points par groupe (pour que chaque proportion moyenne soit précise) et suffisamment de groupes (pour voir la tendance s'il y en a une). Lorsque le nombre d'observations $n > 100$, le nombre de groupes choisi est environ $\sqrt{n}$.

Plot

Le graphique produit par binnedplot indique aussi un intervalle de prédiction à 95% (lignes grises) pour les résidus moyens. Ainsi, si le modèle binomial est bon, environ 95% des résidus devraient se trouver à l'intérieur de cet intervalle. Ici, c'est le cas.

Hosmer-Lemeshow Goodness of Fit

How well our model [mod] fits depends on the difference between the model and the observed data. One approach for binary data is to implement a Hosmer Lemeshow goodness of fit test.

$$\begin{cases} H_0 : & \text{[mod] is adequat,} \\ H_1 : & \text{[mod] is NOT adequat.} \end{cases}$$

```
#library(ResourceSelection)
hoslem.test(train$admit, fitted(mod))
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  train$admit, fitted(mod)
## X-squared = 325, df = 8, p-value < 2.2e-16
```

La *p-value* <0,05 indicates a “bad” fit.

Pseudo-R^2

car le R^2 ne fonctionne pas.

L'estimation des paramètres des modèles linéaires généralisés ne fait pas appel à la méthode des moindres carrés. Pour cette raison, le R^2 basé sur la somme des écarts carrés ne constitue pas une bonne mesure d'ajustement pour ce type de modèle.

La déviance est une mesure d'écart entre les valeurs attendues et les observations calculée à partir de la vraisemblance (L, pour likelihood) du modèle pour les paramètres estimés.

$$D = -2 \log L$$

Pseudo-R^2

Cette expression est aussi égale au premier terme de l'AIC. Plus le modèle accorde une grande probabilité aux observations, plus L

est grand et plus la déviance est petite. Comme pour l'AIC, la valeur absolue de la déviance n'a pas de sens, mais cette quantité est utile pour comparer l'ajustement de différents modèles.

Dans le sommaire du résultat de glm, la déviance du modèle ajusté est indiquée comme Residual Deviance. Le sommaire inclut aussi une autre valeur, Null Deviance, qui correspond à la déviance du modèle nul ne comptant aucun prédicteur. Ces deux valeurs jouent un rôle semblable à la sommes des écarts carrés résiduels et la somme des écarts carrés totaux dans le modèle linéaire. On peut donc définir le pseudo R2 (ou R2 de McFadden) comme la fraction de la déviance du modèle nul expliquée par le modèle incluant les prédicteurs.

Pseudo-R^2

In the following output, we are only interested in the pseudo-R^2 of MacFadden. So here 0.08506168, the model looks bad or poor.

```
pR2(mod)
```

```
## fitting null model for pseudo-r2

##           llh       llhNull            G2      McFadden          r2ML
## -184.99746577 -202.19665227   34.39837298    0.08506168    0.10043246
##          r2CU
##    0.14108581
```

```
pseudo_R2 <- 1 - mod$deviance/mod$null.deviance
```

Pseudo-R^2 de Tjur

La formule du R2 basé sur la déviance s'applique à tous les modèles ajustés par maximum de vraisemblance. Pour la régression logistique spécifiquement, une autre version du coefficient de détermination a été proposée par Tjur:

$$R^2_{Tjur} = \overline{\hat{y}}_{y=1} - \overline{\hat{y}}_{y=0}$$

Autrement dit, le R2 de Tjur mesure la différence entre la réponse moyenne prédite pour les cas où la réponse observée est 1 et la réponse moyenne prédite pour les cas où la réponse observée est 0. Il indique donc à quel point le modèle peut "séparer" les deux groupes $y = 1$ et $y = 0$

A l'extrême, un coefficient de 0 indique que le modèle prédit en moyenne la même réponse pour les deux groupes, tandis qu'un coefficient de 1 indique que le modèle prédit avec certitude la bonne réponse pour toutes les observations.

Pseudo-R^2 de Tjur

```
r2_tjur <- mean(mod$fitted.values[mod$y == 1])
- mean(mod$fitted.values[mod$y == 0])
```

```
## [1] -0.2812251
```

```
r2_tjur
```

```
## [1] 0.3851647
```

Section 2

2. Variables selection

Odds

Definition

The odd for an individual x to obtain the answer $Y = 1$ is defined by

$$odds(x) = \frac{p(x)}{1 - p(x)}, \quad \text{where} \quad p(x) = P(Y = 1/x).$$

Odds ratio (OR)

Definition

The odd ratio (OR) between 2 individuals x and x' is defined by

$$OR(x, x') = \frac{odds(x)}{odds(x')} = \frac{p(x)(1 - p(x'))}{p(x')(1 - p(x))}.$$

Use of OR

The odds ratios make it possible to compare the probabilities of "success" between x and x':

- $OR(x, x') > 1 \Leftrightarrow p(x) > p(x')$,
- $OR(x, x') = 1 \Leftrightarrow p(x) = p(x')$,
- $OR(x, x') < 1 \Leftrightarrow p(x) < p(x')$.

Use of OR

OR can also be interpreted in terms of relative risk. For $p(x)$ and $p(x')$ very small compared to 1 we can approximate the OR by $OR(x, x') \approx p(x)/p(x')$, which allows a relative interpretation between x and x'.

Example

Consider the case of a very rare disease, $Y = 1$ if the disease is present. So if

$$OR(x', x) \sim p(x)/p(x') = 4,$$

indicate the response (illness) is 4 times more likely for individual x than for individual x'.

Other interpretation for OR

Measure of the impact of an explanatory variable. For x and x' in $\mathbb{R}^p$ and for the logistic regression model, we have:

$$\begin{aligned} logit(p(x)) &= \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p, \\ logit(p(x')) &= \beta_0 + \beta_1 x'_1 + \ldots + \beta_p x'_p, \end{aligned}$$

$$\text{Therefore} \qquad OR(x, x') = \exp\left(\beta_1(x_1 - x'_1) + \ldots + \beta_p(x_p - x'_p)\right).$$

To study the influence of an explanatory variable x_j on the odd ratio, we consider 2 observations x and x' such that they differ only on the j th-variable, *i.e.* $x_i = x'_i$ for all $i \neq j$ and $x_j \neq x'_j$. Then the odd ratio in this case is

$$OR(x, x') = \exp\left(\beta_j(x_j - x'_j)\right).$$

Moreover, $x_j - x'_j = 1$ then $OR(x, x') = \exp(\beta_j)$, which allows the study of the influence of the variable j on the response variable without the dependence on x. **These are the latter that R provides** .

OR with R

For a given explanatory variable

- OR=1 means “no effect”.
- OR»1 means “an increase” in the phenomenon studied (here `admission`).
- OR«1 means “an reduction” of the phenomenon studied

Display OR with R

Displays the OR, their confidence interval and their p - *value* (H_0: no effect *vs* H_1: presence of effect)

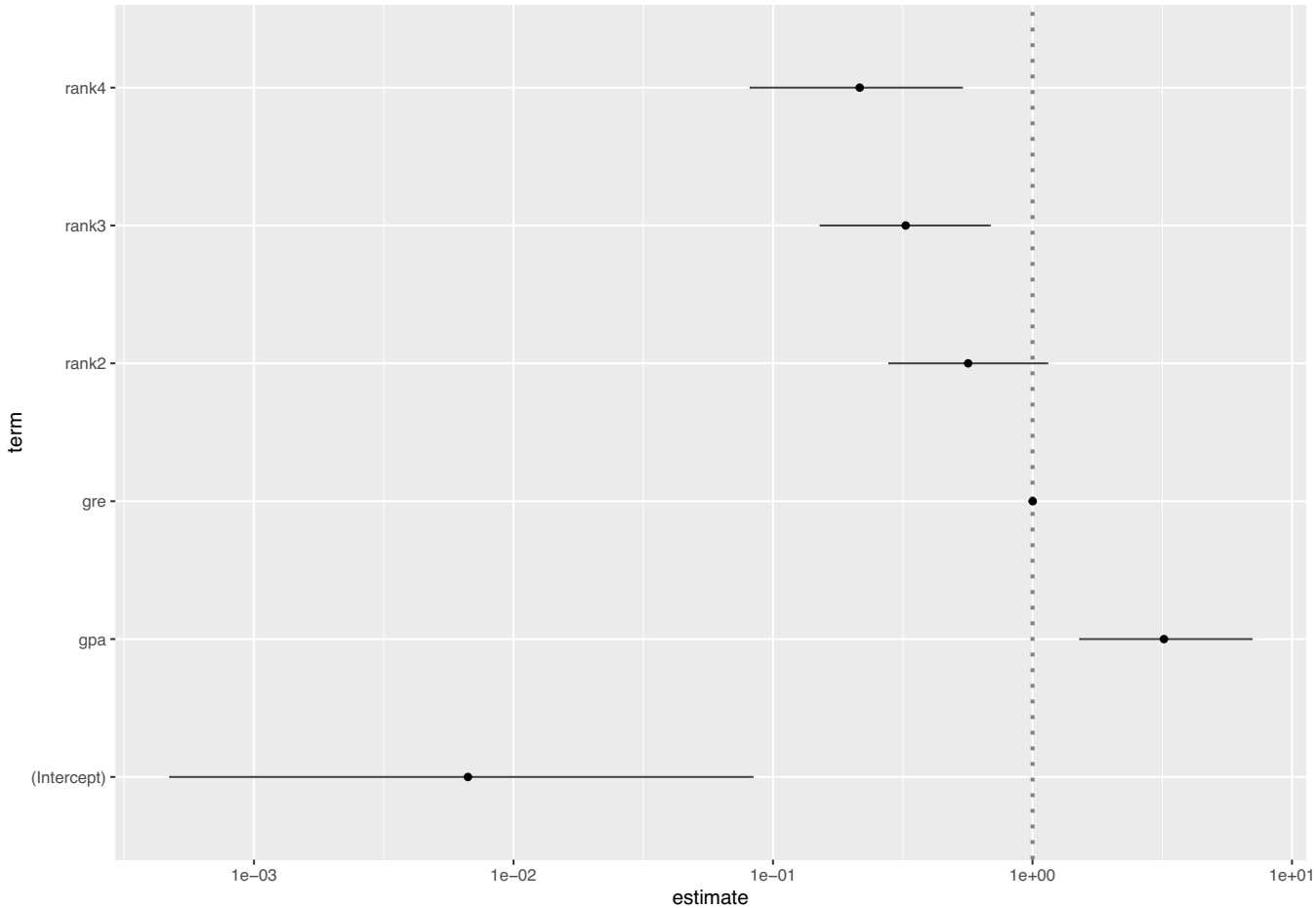
```
odds.ratio(mod)
```

```
##                      OR      2.5 % 97.5 %         p
## (Intercept) 0.00667415 0.00047211 0.0834 0.0001417 ***
## gre         1.00163262 0.99925760 1.0041 0.1801797
## gpa         3.21044159 1.51638084 6.9930 0.0027064 **
## rank2       0.56497364 0.27867718 1.1408 0.1110054
## rank3       0.32454183 0.15149094 0.6844 0.0033315 **
## rank4       0.21589950 0.08132128 0.5354 0.0013219 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

$\Rightarrow$ Here, "gre"can be removed as its OR is close to 1 and its p-*value* is large.

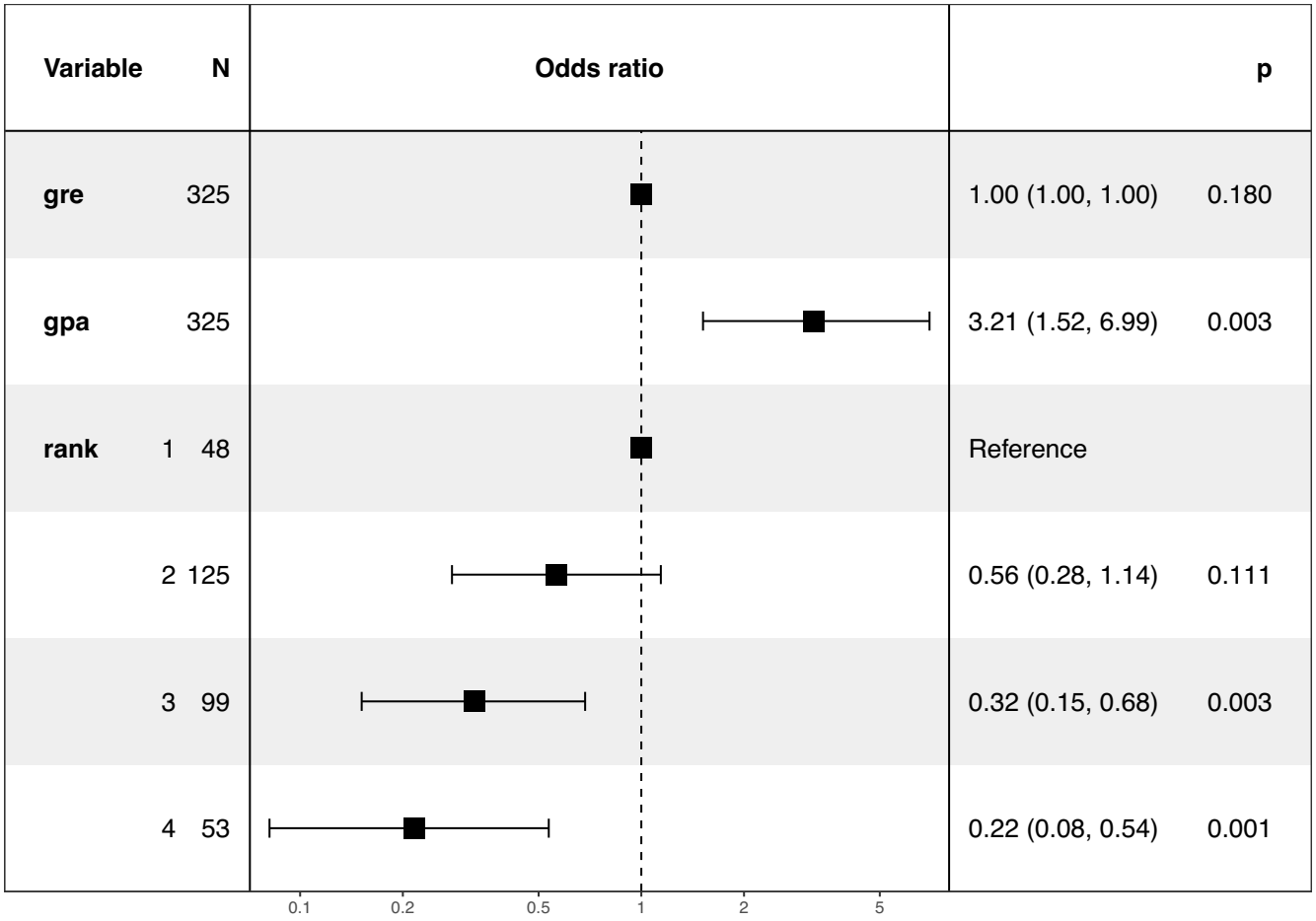
Plot CI for OR with R

```
ggcoef(mod, exponentiate = TRUE)
```



OR with library(forestmodel)

```
forest_model(mod)
```



Step-by-step method

```
step(mod)
```

```
## Start:  AIC=381.99
## admit ~ gre + gpa + rank
##
##        Df Deviance    AIC
## - gre   1   371.81 381.81
## <none>      369.99 381.99
## - gpa   1   379.41 389.41
## - rank  3   384.71 390.71
##
## Step:  AIC=381.81
## admit ~ gpa + rank
##
##        Df Deviance    AIC
## <none>      371.81 381.81
## - rank  3   387.74 391.74
## - gpa   1   387.55 395.55

##
## Call:  glm(formula = admit ~ gpa + rank, family = "binomial", data = train)
##
## Coefficients:
## (Intercept)          gpa        rank2        rank3        rank4
##     -4.7270       1.3735      -0.5712      -1.1645      -1.5642
##
## Degrees of Freedom: 324 Total (i.e. Null);  320 Residual
## Null Deviance:       404.4
## Residual Deviance: 371.8     AIC: 381.8
```

Variables selection

The equivalent of type II tests is the `drop1` function. The latter will in turn remove each variable from the model and perform an analysis of variance to see if the variance changes significantly. (Does not work with `multinom`)

Variables selection

```
drop1(mod, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## admit ~ gre + gpa + rank
##        Df Deviance    AIC     LRT Pr(>Chi)
## <none>      369.99 381.99
## gre     1   371.81 381.81  1.8105 0.178447
## gpa     1   379.41 389.41  9.4141 0.002153 **
## rank    3   384.71 390.71 14.7142 0.002078 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Final model

The previous two methods select the following submodel:

```
modF=glm(admit~gpa+rank,family='binomial',data=train)
ModF=multinom(admit~gpa+rank,data=train)
```

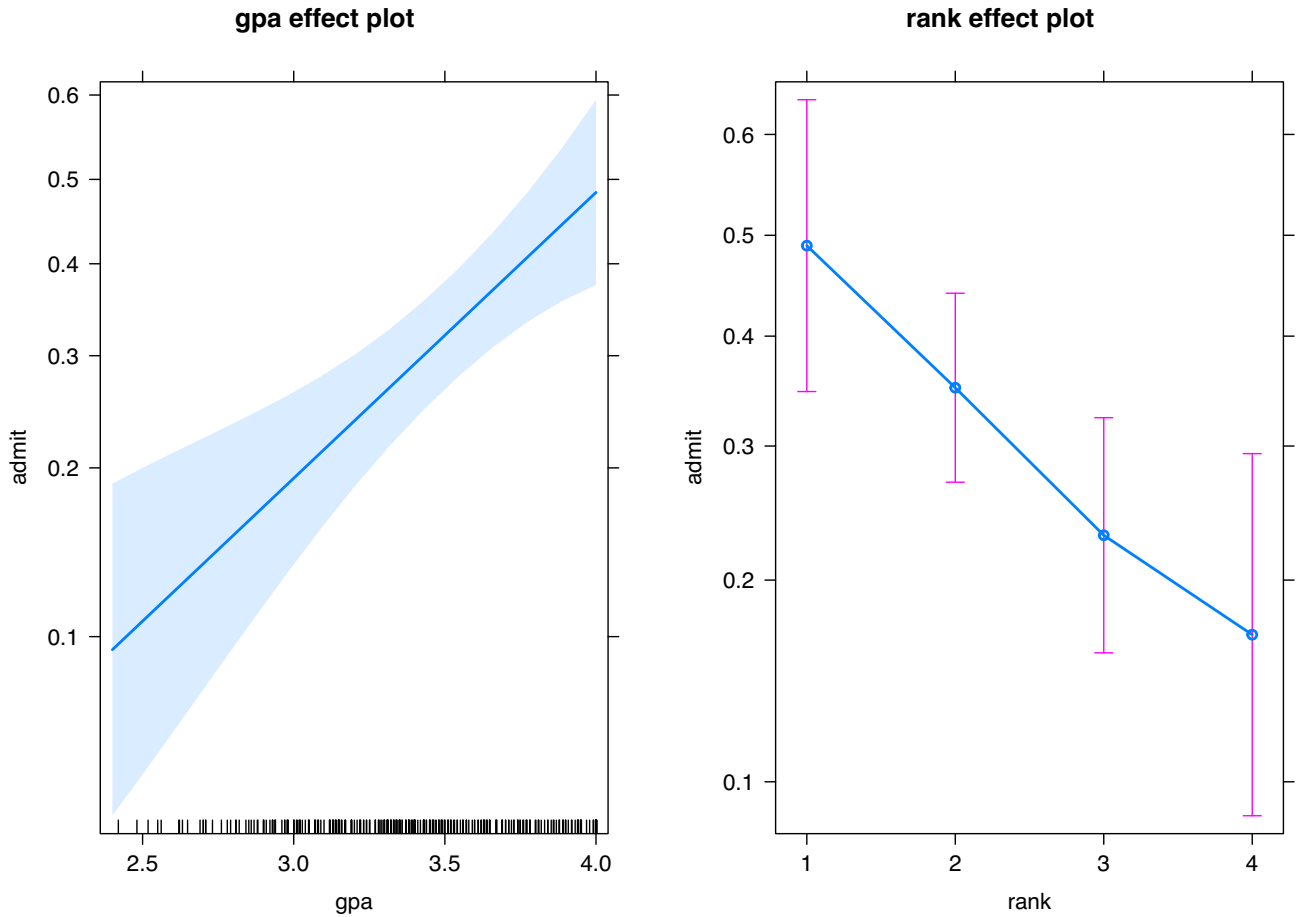
Compare the two models with the anova function.

```
anova(modF, mod, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: admit ~ gpa + rank
## Model 2: admit ~ gre + gpa + rank
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       320     371.81
## 2       319     369.99  1   1.8105   0.1784
```

Plot effects with library(effects):

```
plot(allEffects(modF))
```



Plot effects with library(effects):

- Does not work with `multinom`!
- "`gpa`" has an increasing effect on the admission variable: the larger is the value of "`gpa`", the more one has a chance of be admitted.
- "`rank`": the lower the rank (the better the school of origin) the better the chance of being admitted.

Section 3

3. Confusion Matrix, ROC curve and AUC

Confusion matrix

Recall that the class of positives is class “0” in our example. The **confusion** matrix summurizes:

- The number of True Positive **TP** (predict “0” and the reference is “0”).
- The number of True Negative **TN** (predict “1” and the reference is “1”).
- The number of False Positve **FP** (predict “0” and the reference is “1”).
- The number of False Negative **FN** (predict “1” and the reference is “0”).

These quantities will depend on the threshold s with which we will compare the probabilities estimated by the model.

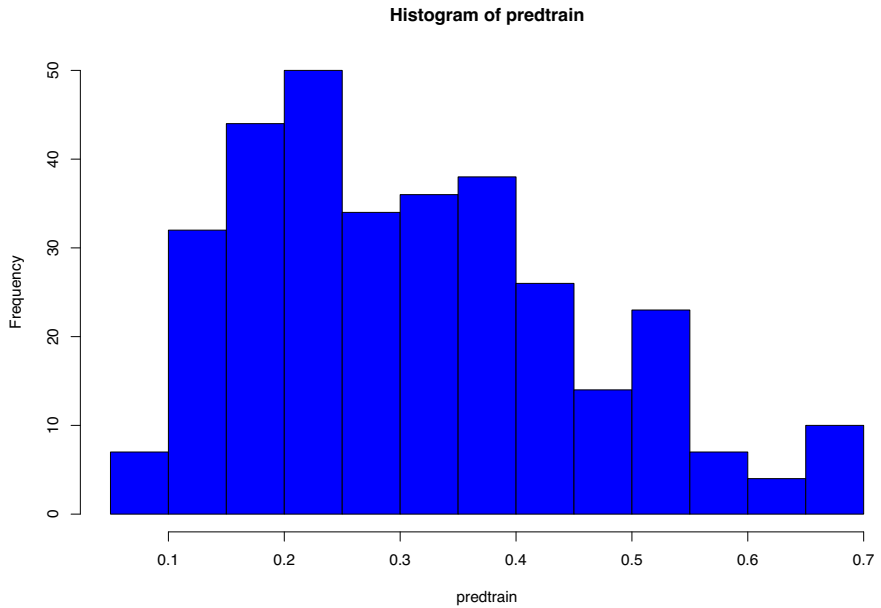
	$Y_i = 0$	$Y_i = 1$
$\widehat{Y}_i = 0$	TP	FP
$\widehat{Y}_i = 1$	FN	TN

Display prediction probabilities

```
#predtrain=fitted(ModF)
predtrain=predict(ModF,train,type="prob")
head(predtrain)
```

```
##         1         2         3         4         6         7
## 0.2822962 0.2992885 0.6828876 0.1290139 0.2354737 0.3466209
```

```
hist(predtrain, col='blue')
```



Display prediction with threshold by default $s = 0.5$

```
p=predict(ModF,train)
head(p)
```

```
## [1] 0 0 1 0 0 0
## Levels: 0 1
```

Display prediction Confusion matrix

```
table(Prediction=p,Reference=train$admit)
```

```
##            Reference
## Prediction   0   1
##          0 208  73
##          1  15  29
```

Several indicators can be deduced from it

- Accuracy : Proportion of correct predictions / classifications

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Sensitivity (TPR) : True Positive Rate compared to actual total positives
  $P = TP + FN$

$$TPR = \frac{TP}{TP + FN} = \frac{TP}{P}$$

- Specificity (TNR) : True Negative Rate compared to actual total negatives
  $N = TN + FP$

$$TNR = \frac{TN}{TN + FP} = \frac{TN}{N}$$

Several indicators can be deduced from it

- FPR : False Positive Rate compared to actual total negatives N

$$FPR = \frac{FP}{TN + FP} = \frac{FP}{N} = 1 - TNR = 1 - specificity$$

- FNR : False Negative Rate compared to actual total positives P

$$FPR = \frac{FN}{TP + FN} = \frac{FN}{P} = 1 - TPR$$

- PPV (*Précision*): Positive Predictive Value compared to estimated total positives $\widehat{P} = TP + FP$

$$PPV = \frac{TP}{TP + FP} = \frac{TP}{\widehat{P}}$$

- NPV : Negative Predictive Value compared to estimated total negatives $\widehat{N} = TN + FN$

$$PNV = \frac{TN}{TN + FN} = \frac{TN}{\widehat{N}}$$

Package caret

```
p=predict(ModF,train)
confusionMatrix(data=p, train$admit)
```

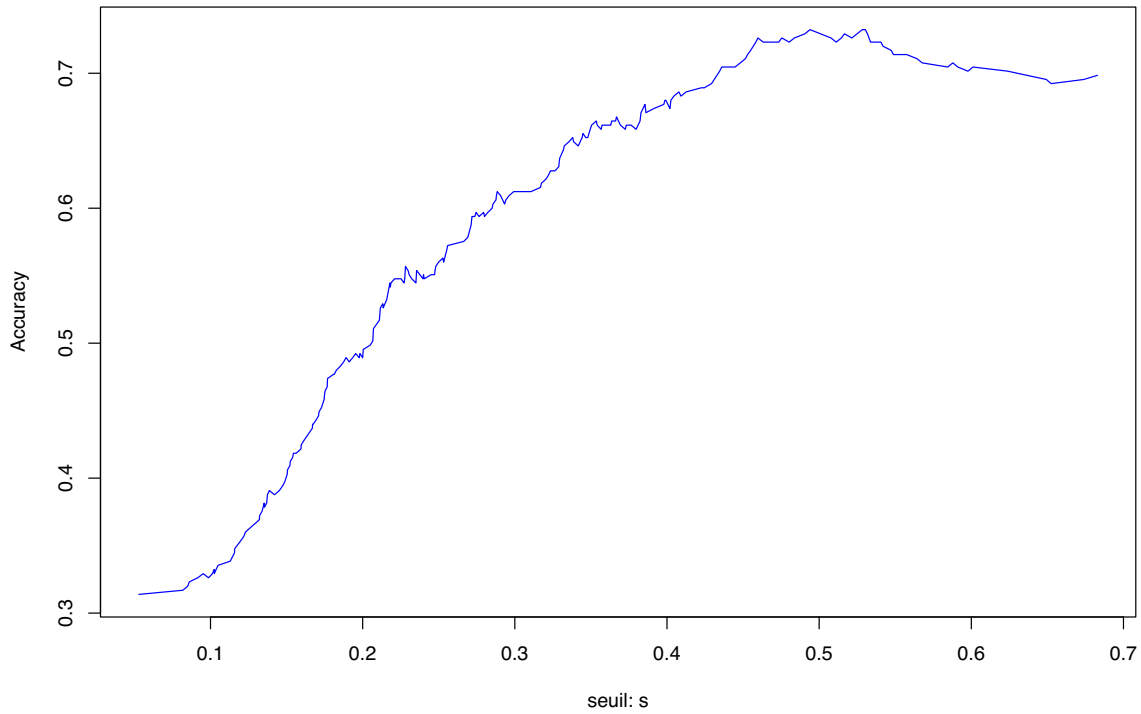
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 208  73
##          1  15  29
##
##                Accuracy : 0.7292
##                  95% CI : (0.6774, 0.7768)
##     No Information Rate : 0.6862
##     P-Value [Acc > NIR] : 0.05189
##
##                   Kappa : 0.2566
##
##  Mcnemar's Test P-Value : 1.23e-09
##
##             Sensitivity : 0.9327
##             Specificity : 0.2843
##          Pos Pred Value : 0.7402
##          Neg Pred Value : 0.6591
##              Prevalence : 0.6862
##          Detection Rate : 0.6400
##    Detection Prevalence : 0.8646
##       Balanced Accuracy : 0.6085
##
##        'Positive' Class : 0
##
```

Accuracy: performance measurement?

The accuracy (0.7292) calculated in the previous frame is for the threshold $s = 0.5$. Let's evaluate the value of the accuracy based on different values of s.

```
prob=predict(ModF,train,type='prob')
p1=prediction(prob,train$admit)
eval=ROCR::performance(p1,"acc")
plot(eval,xlab="seuil: s",col="blue")
```

Accuracy: performance measurement?



Graphically, it is visible that one can improve the accuracy by modifying the threshold. What is the optimal threshold for accuracy?

Accuracy: performance measurement?

```
max=which.max(slot(eval,"y.values")[[1]])
acc_max=slot(eval,"y.values")[[1]][max]
acc_max
```

```
## [1] 0.7323077
```

```
s_opt=slot(eval,"x.values")[[1]][max]
s_opt
```

```
##       235
## 0.5303415
```

By taking a slightly higher threshold, we were able to improve the accuracy (from 0.7292, we went to 0.7323077). But this will be done at the expense of other indicators. Let's take a closer look at this new threshold.

Accuracy: performance measurement?

```
# s=0.5
table(P=p,A=train$admit)
```

```
##    A
## P     0   1
##   0 208  73
##   1  15  29
```

```
prob=predict(ModF,train,type='prob')
# s=0.5303415
pbis=as.factor(ifelse(prob>0.5303415,1,0))
table(P=pbis,A=train$admit)
```

```
##    A
## P     0   1
##   0 213  78
##   1  10  24
```

We have more PV but less NV. The model is less good when it comes to predicting the class of admissions.

Discussion

☛ Accuracy as a performance measure is problematic because it assumes that:

- The distribution of classes is balanced and static.
- All errors have the same cost.

☛ Perhaps it is more interesting to better estimate one modality than another? (* Example: better predict the sick or the healthy individuals? *). The ROC curve makes it possible, among other things, to answer this type of questioning.

☛ A permanent compromise.

ROC (Receiver Operating Characteristic) Curve

☛ To fully assess the performance of a model, we need to analyze both Sensitivity (TPR(s)) and Specificity (TNR(s))for each threshold s.

☛ The ROC is a visualiszation of the different compromise : (1-TNR(s),TPR(s))=(FPR(s),TPR(s))

☛ The ideal procedure (classifier) is $s.t.$

$$\text{(1-TNR(s),TPR(}s\text{))=(FPR(s),TPR(s))=(0,1)}$$

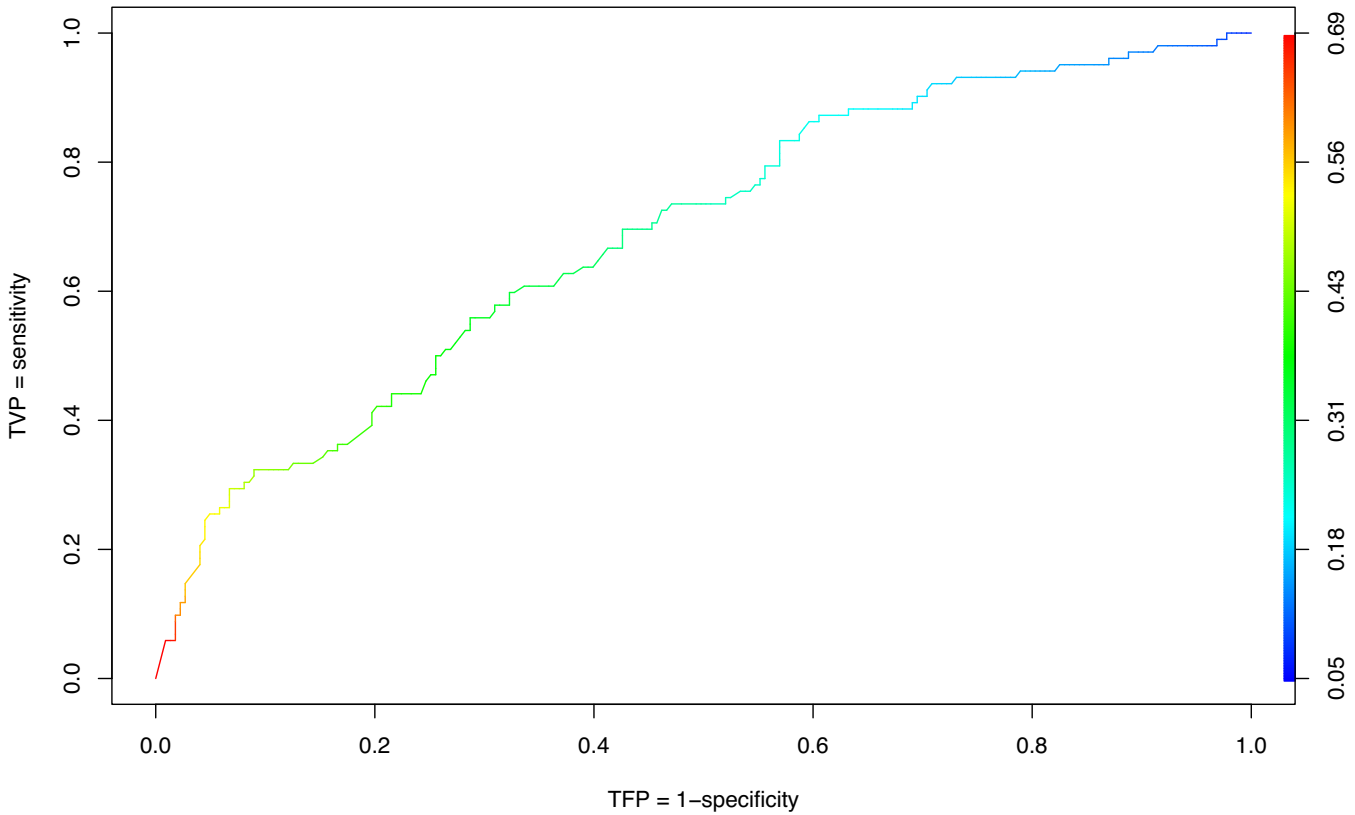
☛ The procedure (1-TNR(s),TPR(s))=(FPR(s),TPR(s))=(0,0) systematically predicts the negative class.

☛ The procedure (1-TNR(s),TPR(s))=(FPR(s),TPR(s))=(1,1) systematically predicts the positive class.

ROC (Receiver Operating Characteristic) Curve

```
prob=predict(ModF,train,type='prob')
p1=prediction(prob,train$admit)
roc=ROCR::performance(p1,"tpr", "fpr")
plot(roc,colorize=TRUE,xlab="TFP = 1-specificity",ylab="TVP = sensitivi
```

ROC (Receiver Operating Characteristic) Curve



The Northwest Rule

Intuitively, a point (FPR(s), TPR(s)) of the ROC curve corresponding to a given classifier is better than another classifier (FPR(s'), TPR(s')) if it is northwest of it, *i.e.* it will have a higher TPR and a lower FPR.

AUC (Area Under Curve)

Each ROC curve is characterized by its area under the curve: AUC.

- It is an aggregated measure of performance for all possible classification thresholds.
- AUC values are between 0 (all predictions are wrong) and 1 (all predictions are correct).

```
prob=predict(ModF,train,type='prob')
p1=prediction(prob,train$admit)
auc=ROCR::performance(p1,"auc")
auc=unlist(slot(auc,"y.values"))
auc
```

```
## [1] 0.6874615
```

AUC advantages

- AUC is scale invariant. It measures how well predictions rank, rather than their absolute values.
- AUC is independent of classification thresholds. It measures the quality of the precision of the model regardless of the classification threshold selected.

AUC limitations

- Scale invariance is not always desirable. For example, sometimes we need to get precisely calibrated probabilities, which the AUC cannot determine.
- Independence from classification thresholds is also not always desirable. If the cost generated by the FN and the cost generated by the FP are not the same, it may be important to minimize one of the 2 types of classification errors.
    - *For example: we prefer to minimize, in the context of a serious and contagious disease, in priority false negatives*.

AUC is not a criterion to be used for this type of optimization.

The test : confusion matrix

```
p_t=predict(ModF,test)
confusionMatrix(data=p_t, test$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48 20
##          1  2  5
##
##                Accuracy : 0.7067
##                  95% CI : (0.5902, 0.8062)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.2731492
##
##                   Kappa : 0.1951
##
##  Mcnemar's Test P-Value : 0.0002896
##
##             Sensitivity : 0.9600
##             Specificity : 0.2000
##          Pos Pred Value : 0.7059
##          Neg Pred Value : 0.7143
##              Prevalence : 0.6667
##          Detection Rate : 0.6400
##    Detection Prevalence : 0.9067
##       Balanced Accuracy : 0.5800
##
##        'Positive' Class : 0
##
```

The test : AUC

```
prob_t=predict(ModF,test,type='prob')
p1_t=prediction(prob_t,test$admit)
auc_t=ROCR::performance(p1_t,"auc")
auc_t=unlist(slot(auc_t,"y.values"))
auc_t
```

```
## [1] 0.6436
```

ROC curves : Train/test

```
prob_t=predict(ModF,test,type='prob')
p1_t=prediction(prob_t,test$admit)
roc_t=ROCR::performance(p1_t,"tpr", "fpr")

par(mfrow = c(1, 2))
plot(roc,colorize=TRUE,main = "ROC train, AUC=0.6874615",
     xlab="TFP = 1-specificity",ylab="TVP = sensitivity")
plot(roc_t,colorize=TRUE,main = "ROC test, AUC=0.643625",
     xlab="TFP = 1-specificity",ylab="TVP = sensitivity")
```

ROC curves : Train/test