Survival and longitudinal data analysis

Comparaison de la prédiction par modèle de survie et par algorithmes de classification

Guedj O. et Marcoux Pépin T.

06/11/2019

Contents

1. Packages **2**

2. Données **2**
2.1. Importation des données et mise en forme . . . 2
2.2. Gestion des valeurs manquantes . . . 3
2.3. Data Exploration . . . 3
2.3.1. Rechute . . . 3
2.3.2. Variables décrivant la tumeur . . . 3
2.4. Scaling . . . 3
2.5. Variable de censure . . . 6
2.6. Split des données . . . 6

3. Modèles de Survie **7**
3.1. Kaplan Meier global . . . 7
3.2. Modèle de Cox full . . . 8
3.2.1 Modèle . . . 8
3.2.2 Prédiction, ROC et AUC . . . 10
3.2.3 Diagnostique du modèle . . . 12
3.3 Step AIC . . . 17
3.3.1 Cox model . . . 17
3.3.2 Prediction, ROC et AUC . . . 17
3.3.3 Diagnostique du modèle . . . 20

4. Classification **24**
4.1 Création du train et du test . . . 24
4.1.1 Avec toutes les variables . . . 24
4.1.2 Avec les variables issues de l'AIC . . . 25
4.2 Prédictions, ROC, AUC et Accuracy sur les variables issues de l'AIC . . . 25
4.2.1 Naives Bayes . . . 25
4.2.2 Linear Discriminant Analysis . . . 25
4.2.3 Quadratic Discriminant Analysis . . . 26
4.2.4. Support Vector Machine . . . 26
4.2.5. kNN . . . 26
4.3. Prédictions, ROC, AUC et Accuracy sur les variables issues du modèle complet . . . 28
4.3.1 Naives Bayes . . . 28
4.3.2 Linear Discriminant Analysis . . . 29
4.3.3 Quadratic Discriminant Analysis . . . 29
4.3.4. Support Vector Machine . . . 29
4.3.5. kNN . . . 30
4.3.6. Random Forest . . . 30

5. Résumé des performances des modèles **31**

6. Conclusion **33**

1. Packages

```
suppressMessages(library(KMsurv))
suppressMessages(library(tidyverse))
suppressMessages(library(survival))
suppressMessages(library(dplyr))
suppressMessages(library(survminer))
suppressMessages(library(ggplot2))
suppressMessages(library(ggfortify))
suppressMessages(library(caTools))
suppressMessages(library(caret))
suppressMessages(library(MASS))
suppressMessages(library(psych))
suppressMessages(library(kableExtra))
suppressMessages(library(pROC))
suppressMessages(library(e1071))
suppressMessages(library(class))
suppressMessages(library(randomForest))
```

2. Données

Les données étudiées sont issues du jeu de données wpcb: une étude portée sur les cancer du sein de 198 femmes du Winsconsis, USA. Le jeu de données est composé de 198 individus décrits par 35 variables dont: * Une variable d'identification `id`, * Une variable indiquant s'il y a eu rechute du cancer ou non `recurrent`, * Une variable indiquant le temps de rechute `time` (en mois) pour les individus ayant rechuté et indiquant le temps à l'état sain pour les individus n'ayant pas rechuté, * Le reste des variables décrivent la tumeur : sa texture, son périmètre, sa surface...

2.1. Importation des données et mise en forme

Dans le dataset, les NA sont représentées par des "?".

```
data = read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wpbc
                header=F,
                sep=",",
                na = "?")
```

De plus les variables sont nommées de `var1` à `var35`, il faut donc leur donner des noms plus appropriés.

```
var.names = paste0(rep(c('radius','texture','perimeter','area','smoothness','compactness',
                         'concavity','concave points','symmetry','fractal dimension'),3),
                   c(rep('_mean',10),rep('_SD',10),rep('_worst',10)))
names(data) = c('id','recurrent','time', var.names,c('Tumor size','Lymph node status'))
```

On transforme le type de l'identifiant en factor. Dans la variable `recurrent` on remplace les N par `FALSE` et les R par `TRUE`. On transforme également la variable `time` en numeric.

```
data = data %>% mutate(id = factor(id)) %>%
                mutate( recurrent = recode_factor(recurrent , "N" = FALSE, 'R' = TRUE ))

data$recurrent = as.logical(data$recurrent)
```

```
data = dplyr::mutate(data,time=as.numeric(time))
```

2.2. Gestion des valeurs manquantes

```
na.nbr = NULL
for( i in colnames(data)){
  na.nbr[i] = length(which(is.na(data[,i])))
}
rm(i)
tab.na = cbind(names(data), na.nbr)
kable(tab.na, col.names = c("Variables","Nombre de NA"), row.names = F,
      caption = "Vue d'ensemble des données manqanutes des données wpbc") %>%
  kable_styling(latex_options = "striped", stripe_color = "gray!6")
```

(voir Table 1)

On fait le choix de remplacer les NA de `Lymph node status` par la médiane de la variable correspondante.

```
data.no.NA = data %>% replace_na(list(`Lymph node status` = median(data$`Lymph node status`,na.rm =T)))
```

2.3. Data Exploration

2.3.1. Rechute

```
tab.recur = cbind(c("Non","Oui"),table(data.no.NA$recurrent), round(prop.table(table(data.no.NA$recurre
kable(tab.recur, col.names = c("Rechute ","n","%"),
      row.names = F,
      caption = "Nombre de rechutes") %>%
  kable_styling(latex_options = "basic")
```

On remarque la faible proportion de rechutes (Table 2). C'est un élément auquel il faut préter attention, d'une part lors du split des données et d'autre part pour expliquer une eventuelle difficulté des algorithmes de machine learning à prédire cette rechute.

2.3.2. Variables décrivant la tumeur

```
desc.var.tum = round(describe(data.no.NA[,4:35])[c(1,2,3,4,5,8,9)],2)
kable(desc.var.tum, caption = "Aperçu des variables décrivants la tumeur") %>%
  kable_styling(latex_options = "striped", stripe_color = "gray!6")
```

(Table 3)

2.4. Scaling

Pour pouvoir comparer les variables entre elles on les normalise en centrant et réduisant les données (sauf les 3 premières variables: `id`, `recurrent` et `time`).

```
scale = function(x){
  (x- mean(x,na.rm=T)) / sd(x,na.rm=T)
}
```

Table 1: Vue d'ensemble des données manqanutes des données wpbc

Variables	Nombre de NA
id	0
recurrent	0
time	0
radius_mean	0
texture_mean	0
perimeter_mean	0
area_mean	0
smoothness_mean	0
compactness_mean	0
concavity_mean	0
concave points_mean	0
symmetry_mean	0
fractal dimension_mean	0
radius_SD	0
texture_SD	0
perimeter_SD	0
area_SD	0
smoothness_SD	0
compactness_SD	0
concavity_SD	0
concave points_SD	0
symmetry_SD	0
fractal dimension_SD	0
radius_worst	0
texture_worst	0
perimeter_worst	0
area_worst	0
smoothness_worst	0
compactness_worst	0
concavity_worst	0
concave points_worst	0
symmetry_worst	0
fractal dimension_worst	0
Tumor size	0
Lymph node status	4

Table 2: Nombre de rechutes

Rechute	n	%
Non	151	76.26
Oui	47	23.74

Table 3: Aperçu des variables décrivants la tumeur

	vars	n	mean	sd	median	min	max
radius_mean	1	198	17.41	3.16	17.29	10.95	27.22
texture_mean	2	198	22.28	4.30	21.75	10.38	39.28
perimeter_mean	3	198	114.86	21.38	113.70	71.90	182.10
area_mean	4	198	970.04	352.15	929.10	361.60	2250.00
smoothness_mean	5	198	0.10	0.01	0.10	0.07	0.14
compactness_mean	6	198	0.14	0.05	0.13	0.05	0.31
concavity_mean	7	198	0.16	0.07	0.15	0.02	0.43
concave points_mean	8	198	0.09	0.03	0.09	0.02	0.20
symmetry_mean	9	198	0.19	0.03	0.19	0.13	0.30
fractal dimension_mean	10	198	0.06	0.01	0.06	0.05	0.10
radius_SD	11	198	0.60	0.31	0.53	0.19	1.82
texture_SD	12	198	1.26	0.53	1.17	0.36	3.50
perimeter_SD	13	198	4.26	2.19	3.77	1.15	13.28
area_SD	14	198	70.23	47.98	58.45	13.99	316.00
smoothness_SD	15	198	0.01	0.00	0.01	0.00	0.03
compactness_SD	16	198	0.03	0.02	0.03	0.01	0.14
concavity_SD	17	198	0.04	0.02	0.04	0.01	0.14
concave points_SD	18	198	0.02	0.01	0.01	0.01	0.04
symmetry_SD	19	198	0.02	0.01	0.02	0.01	0.06
fractal dimension_SD	20	198	0.00	0.00	0.00	0.00	0.01
radius_worst	21	198	21.02	4.24	20.52	12.84	35.13
texture_worst	22	198	30.14	6.02	30.13	16.67	49.54
perimeter_worst	23	198	140.35	28.89	136.50	85.10	232.20
area_worst	24	198	1404.96	586.01	1295.00	508.10	3903.00
smoothness_worst	25	198	0.14	0.02	0.14	0.08	0.22
compactness_worst	26	198	0.37	0.16	0.35	0.05	1.06
concavity_worst	27	198	0.44	0.17	0.40	0.02	1.17
concave points_worst	28	198	0.18	0.05	0.18	0.03	0.29
symmetry_worst	29	198	0.32	0.08	0.31	0.16	0.66
fractal dimension_worst	30	198	0.09	0.02	0.09	0.06	0.21
Tumor size	31	198	2.85	1.94	2.50	0.40	10.00
Lymph node status	32	198	3.17	5.43	1.00	0.00	27.00

Table 4: Variable de censure

z	n
0	129
1	29
NA	40

```
data.scaled =  data.no.NA %>% mutate_at(names(data)[-c(1,2,3)], scale)
```

2.5. Variable de censure

On note Z la variable de censure. Trois cas sont possibles:

- Si le temps d'observation est plus petit ou égal à 24 mois et qu'il y a eu rechute alors il a censure et Z = 1
- Si le temps d'observation est superieur à 24 mois et qu'il y a eu rechute alors il n'y a pas censure et Z = 0
- Enfin, si le temps d'observation est superieur à 24 mois et qu'il n'y a pas eu rechute alors on ne sait pas et Z = NA

```
data.scaled = data.scaled %>%
  mutate(Z = ifelse( (time <= 24) & (recurrent == T),1,
                  ifelse( (time > 24) & (recurrent == T),0,
                          ifelse( (time > 24) & (recurrent == F),0,NA)
                          )
                  )
         )
data.scaled$Z = as.factor(data.scaled$Z)

kable(table(data.scaled$Z, useNA = "always"),
      caption = "Variable de censure",
      col.names = c("z","n"))
```

(voir Table 4)

2.6. Split des données

On ajoute à data.scaled une variable d'indices: chiffres de 1 à 198 afin d'effectuer un tirage aléatoire d'invidus dans le jeu de données.

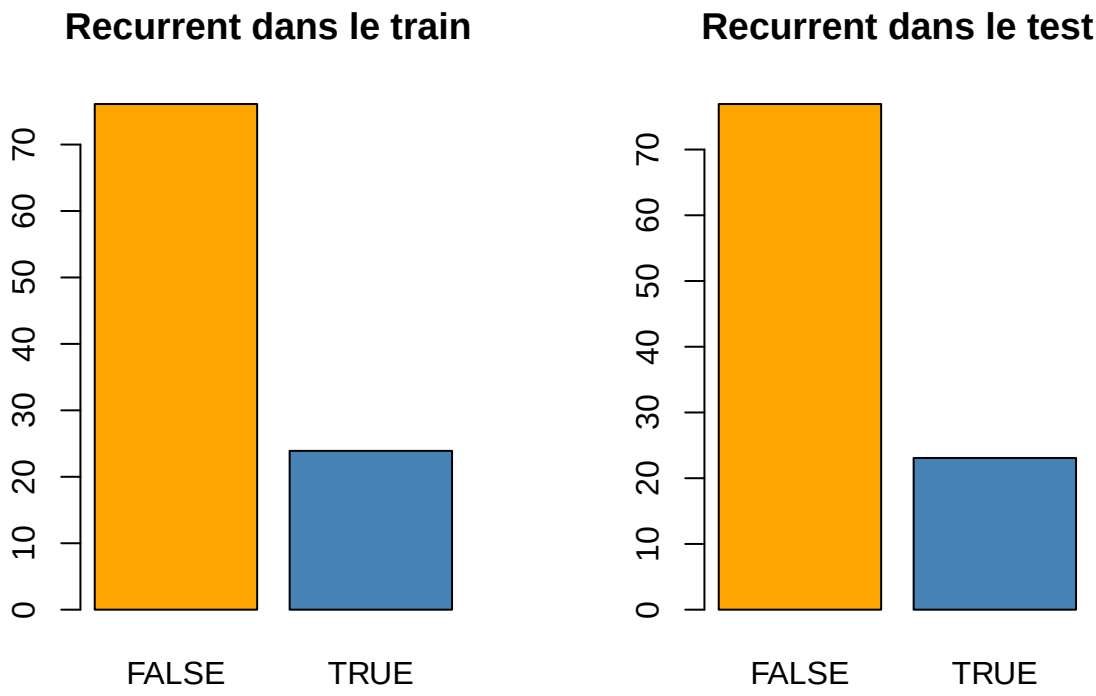
Puis on utilise la fonction `createDataPartition` pour attribuer 80% des données au train et les stocker dans une matrice (list = FALSE).

```
set.seed(42) #Create simulated values that are reproductible.
data.scaled = data.scaled %>% mutate(id_1n = c(1:nrow(data.scaled)))
trainIndex = createDataPartition(data.scaled$recurrent, p = 0.8, list = FALSE, times = 1)
train = data.scaled %>% filter(id_1n %in% trainIndex)
test = data.scaled[-which(data.scaled$id_1n %in% train$id_1n),]
```

On peut vérifier graphiquement la bonne stratification des données:

```
par(mfrow = c(1,2))
barplot(prop.table(table(train$recurrent))*100, col = c("orange","steelblue"),
```

```
        main = "Recurrent dans le train ")
barplot(prop.table(table(test$recurrent))*100, col = c("orange","steelblue"),
        main = "Recurrent dans le test")
```



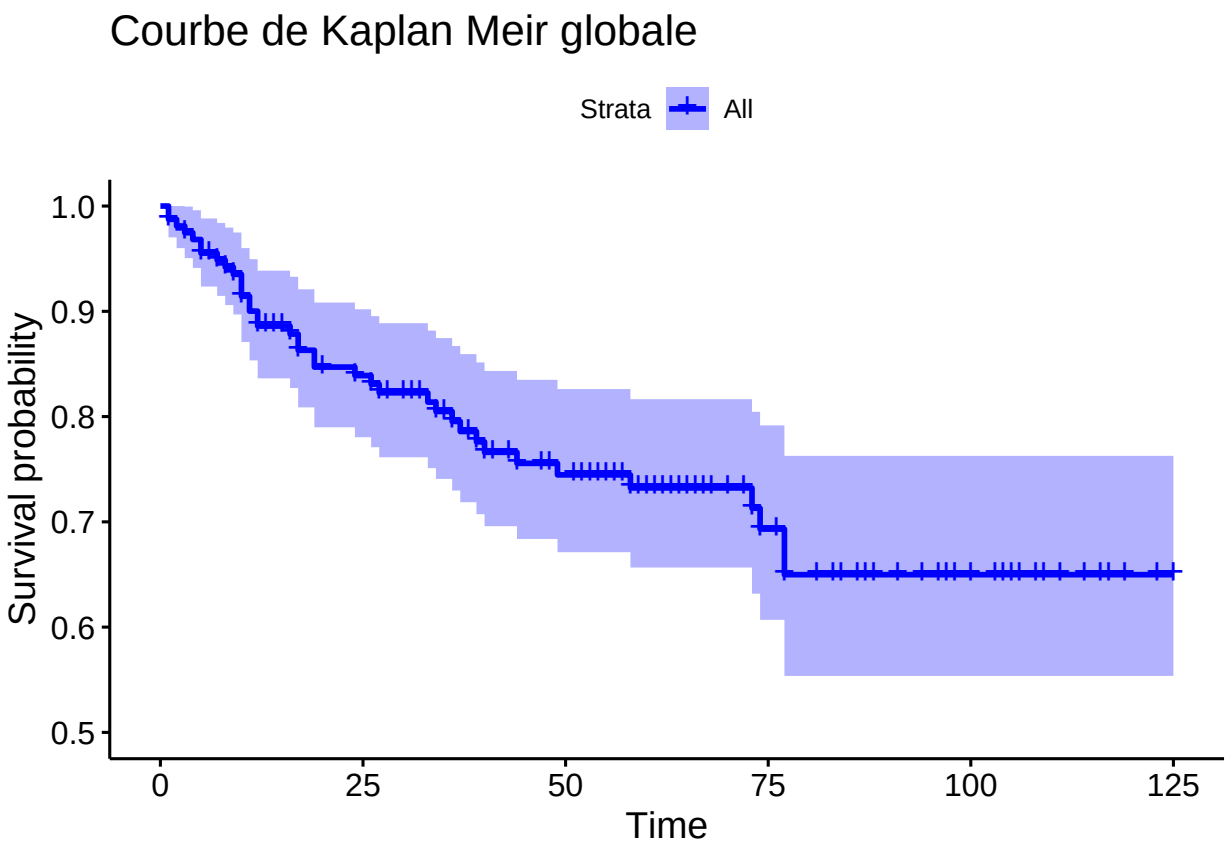
Enfin, on crée une autre partition des données en train et en test (80% / 20%) sans la variable de censure:

```
train.surv = train %>% dplyr::select(-Z)
test.surv = test %>% dplyr::select(-Z)
```

3. Modèles de Survie

3.1. Kaplan Meier global

```
km.global = survfit(Surv(time, recurrent) ~ 1, data = train.surv)
ggsurvplot(km.global, ylim = c(0.5,1), palette = "blue") + ggtitle("Courbe de Kaplan Meir globale")
```



3.2. Modèle de Cox full

3.2.1 Modèle

On commence par un modèle full (sans les deux variables d'identification):

```
cox.all = coxph(Surv(time,recurrent)~.-id -id_1n ,data = train.surv)

df.cox.all = data.frame(format(summary(cox.all)$conf.int[,1],scientific = T, digit = 2),
                        paste("[",format(summary(cox.all)$conf.int[,3], scientific = T, digit = 2),
                              ";",format(summary(cox.all)$conf.int[,4],scientific = T, digit = 2),"]"),
                        format(summary(cox.all)$coefficient[,5], scientific = T, digit = 2),
                        ifelse(summary(cox.all)$coefficient[,5] <= 0.05, "*","")
)

colnames(df.cox.all) = c("exp(coef)", "IC", "p", "Significativité ")
kable(df.cox.all, caption = "Modèle de cox avec toutes les covariables") %>%
  kable_styling(latex_options = "striped", stripe_color = "gray!6")
```

On remarque que très peu de variables sont significatives.

```
cat("La probabilité de ne pas rechuter à 24 mois est de",
    round(unlist(summary(km.global, time = 24))$surv, 2),"avec un IC de [",
    round(unlist(summary(km.global, time = 24))$lower,2),";" ,
    round(unlist(summary(km.global, time = 24))$upper, 2), "]")
```

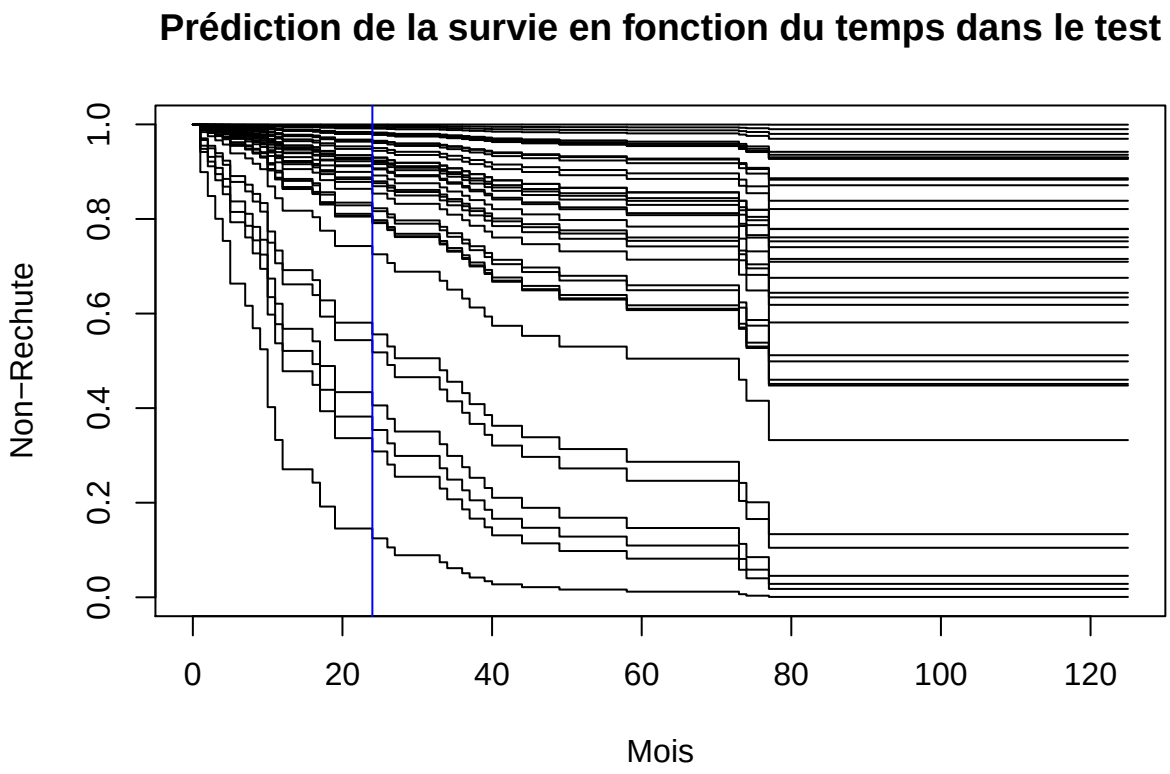
Table 5: Modèle de cox avec toutes les covariables

	exp(coef)	IC	p	Significativité
radius_mean	6.1e-08	[4.0e-16 ; 9.2e+00]	8.4e-02	
texture_mean	4.7e-01	[1.1e-01 ; 2.0e+00]	3.1e-01	
perimeter_mean	1.2e+06	[8.5e-03 ; 1.7e+14]	1.4e-01	
area_mean	5.8e+01	[1.5e-01 ; 2.3e+04]	1.8e-01	
smoothness_mean	4.1e+00	[9.8e-01 ; 1.7e+01]	5.3e-02	
compactness_mean	4.2e+00	[1.6e-01 ; 1.1e+02]	3.9e-01	
concavity_mean	1.7e-01	[1.1e-02 ; 2.6e+00]	2.0e-01	
‘concave points_mean‘	5.9e-01	[6.2e-02 ; 5.5e+00]	6.4e-01	
symmetry_mean	5.1e-01	[1.9e-01 ; 1.4e+00]	1.8e-01	
‘fractal dimension_mean‘	8.9e-02	[1.1e-02 ; 7.1e-01]	2.3e-02	*
radius_SD	4.3e-01	[3.8e-03 ; 4.9e+01]	7.3e-01	
texture_SD	3.4e-01	[8.8e-02 ; 1.3e+00]	1.1e-01	
perimeter_SD	7.0e+00	[7.6e-02 ; 6.5e+02]	4.0e-01	
area_SD	9.5e-01	[5.7e-02 ; 1.6e+01]	9.7e-01	
smoothness_SD	1.0e+00	[2.9e-01 ; 3.8e+00]	9.5e-01	
compactness_SD	1.0e+01	[1.1e+00 ; 1.0e+02]	4.5e-02	*
concavity_SD	1.9e-01	[2.0e-02 ; 1.8e+00]	1.5e-01	
‘concave points_SD‘	1.2e+00	[3.0e-01 ; 4.6e+00]	8.2e-01	
symmetry_SD	8.9e-01	[2.7e-01 ; 2.9e+00]	8.5e-01	
‘fractal dimension_SD‘	8.4e-01	[1.4e-01 ; 5.1e+00]	8.5e-01	
radius_worst	5.9e+01	[6.8e-03 ; 5.1e+05]	3.8e-01	
texture_worst	2.2e+00	[3.4e-01 ; 1.5e+01]	4.0e-01	
perimeter_worst	9.2e-02	[1.2e-04 ; 7.2e+01]	4.8e-01	
area_worst	8.3e-02	[1.4e-04 ; 5.0e+01]	4.5e-01	
smoothness_worst	1.8e+00	[5.5e-01 ; 6.0e+00]	3.2e-01	
compactness_worst	3.7e-02	[2.5e-03 ; 5.5e-01]	1.6e-02	*
concavity_worst	6.8e+00	[6.9e-01 ; 6.8e+01]	1.0e-01	
‘concave points_worst‘	7.1e-01	[1.8e-01 ; 2.8e+00]	6.3e-01	
symmetry_worst	1.5e+00	[3.2e-01 ; 7.2e+00]	6.1e-01	
‘fractal dimension_worst‘	3.4e+00	[3.1e-01 ; 3.8e+01]	3.1e-01	
‘Tumor size‘	9.1e-01	[6.0e-01 ; 1.4e+00]	6.5e-01	
‘Lymph node status‘	1.6e+00	[1.1e+00 ; 2.4e+00]	1.0e-02	*

```
## La probabilité de ne pas rechuter à 24 mois est de 0.84 avec un IC de [ 0.78 ; 0.9 ]
```

3.2.2 Prédiction, ROC et AUC

```
plot(survfit(cox.all , newdata = test.surv), xlab = "Mois", ylab="Non-Rechute",
     main = "Prédiction de la survie en fonction du temps dans le test")
abline(v = 24, col = "blue")
```




```
pred.cox.all = survfit(cox.all , newdata = test.surv)
prob.pred.24.all = pred.cox.all$surv[24,]
```

`prob.pred.24` contient les predictions de survie à 24 mois des individus du jeu de données test.

Une probabilité élévée indique une grande chance de survie donc une faible chance d'avoir une rechute de cancer.

En fixant un seuil à 0.5 on peut ainsi dire que, si un individu a une propabilité de survie plus petite que 0.5 alors il a une "grande" chance de rechuter. Donc, on attribue la valeur 1 à la variable de rechute prédite et 0 dans le cas contraire.

```
pred.24.all = ifelse(prob.pred.24.all <= 0.5, 1, 0)
pred.24.all = as.factor(pred.24.all)
```

On obtient alors la matrice de confusion suivante (Table 6):

```
kable(table(Predicted = pred.24.all,Real = test.surv$recurrent), caption = "Matrice de confusion pour la
```

Table 6: Matrice de confusion pour la prédiction dans le modele cox.all

	Real.0	Real.1
0	26	8
1	4	1

Le modèle a du mal à prédire correctement les cas de rechute. Cela est peut-être du au faible nombre de rechutes dans le jeu de données par rapport à celui de non-rechutes.

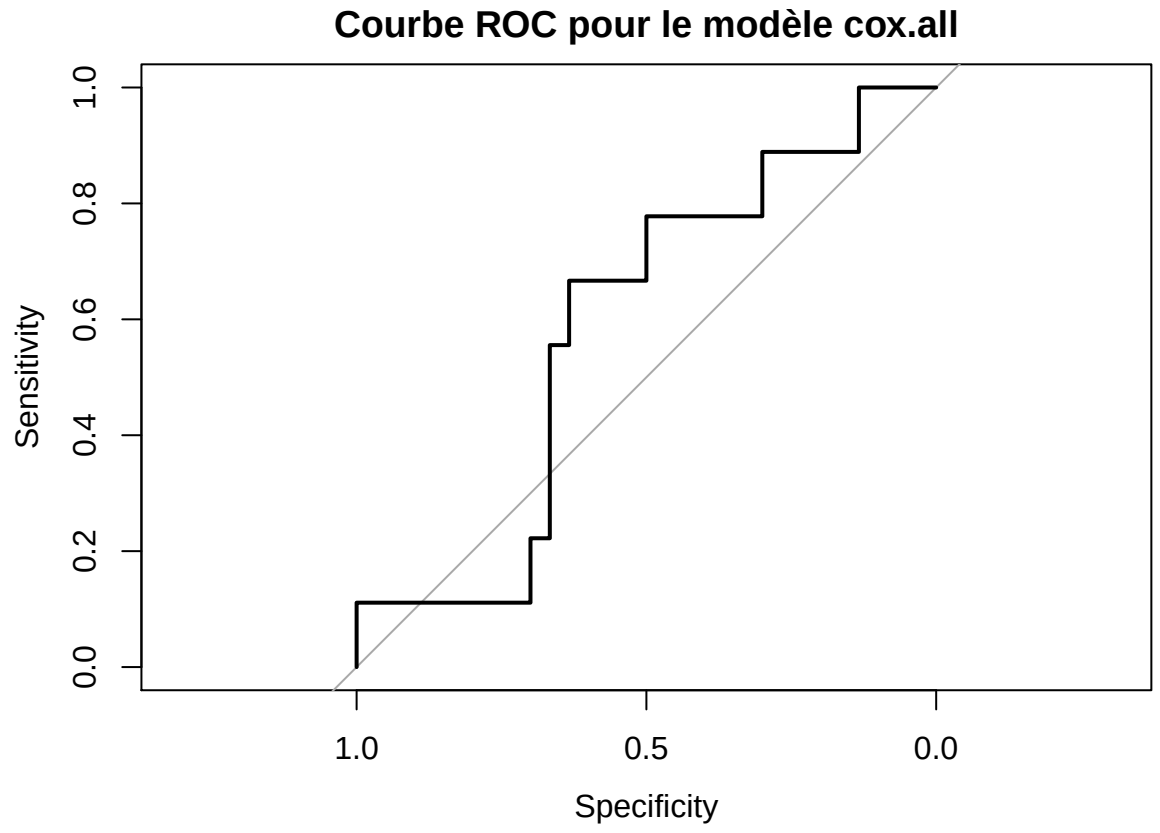
En faisant varier le seuil précédent entre 0 et 1 on obtient la courbe ROC suivante:

```
roc.cox.all = roc(response = test.surv$recurrent, predictor = prob.pred.24.all)
```

```
## Setting levels: control = FALSE, case = TRUE
```

```
## Setting direction: controls > cases
```

```
plot(roc.cox.all, main = "Courbe ROC pour le modèle cox.all")
```




```
auc.cox.all = auc(roc.cox.all)
cat("L'AUC du modèle cox.all est de:",auc.cox.all)
```

```
## L'AUC du modèle cox.all est de: 0.5851852
```

A seuil fixé on peut calculer une précision du modèle, afin de comparer ces résultats aux modèles dont ont ne peut pas tracer la courbe ROC.

Etant donné qu'il est très important de bien classifier les individus qui vont effectivement faire une rechute à 24 mois, on choisi un seuil "bas".

```r
pred.24.all.bis = ifelse(prob.pred.24.all <= 0.5, 1, 0)
pred.24.all.bis = as.factor(pred.24.all.bis)
table(Pred = pred.24.all.bis, real = test.surv$recurrent)
```

```
##     real
## Pred FALSE TRUE
##    0    26    8
##    1     4    1
```

On a donc une précision de $ (26+1)/39 = 0.6923077$:

3.2.3 Diagnostique du modèle

Le modèle de Cox a une hypothèse importante: la proportionnalité des risques relatifs (équivalente à la non dépendance en temps des résidus de Schonfeld).

Pour la vérifier on peut:

- Faire un test
- Regarer l'allure des résidus

```r
test.hyp.cox.all <- cox.zph(cox.all)
kable(format(as.matrix(test.hyp.cox.all$table), scientific = T, digit = 2)) %>%
  kable_styling(latex_options = "striped", stripe_color = "gray!6")
```

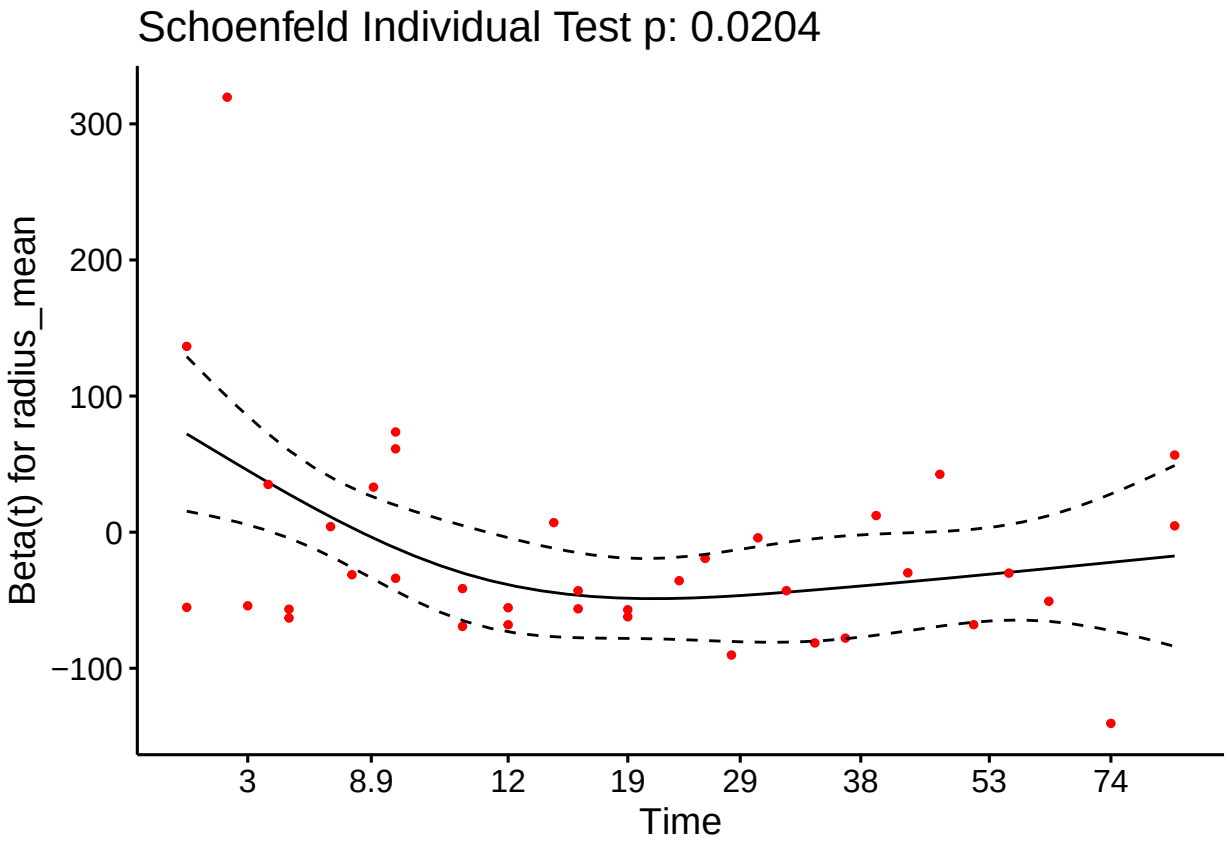
	rho	chisq	p
radius_mean	-2.9e-01	5.4e+00	2.0e-02
texture_mean	-1.2e-01	5.5e-01	4.6e-01
perimeter_mean	2.5e-01	4.3e+00	3.9e-02
area_mean	2.8e-01	2.9e+00	9.0e-02
smoothness_mean	1.7e-01	1.5e+00	2.2e-01
compactness_mean	-2.8e-01	7.4e+00	6.6e-03
concavity_mean	-5.3e-02	1.1e-01	7.4e-01
‘concave points_mean‘	-1.8e-01	9.9e-01	3.2e-01
symmetry_mean	4.3e-01	1.0e+01	1.3e-03
‘fractal dimension_mean‘	1.6e-01	1.5e+00	2.1e-01
radius_SD	1.1e-01	5.4e-01	4.6e-01
texture_SD	-1.1e-02	6.6e-03	9.4e-01
perimeter_SD	5.4e-03	1.3e-03	9.7e-01
area_SD	-2.1e-01	1.8e+00	1.9e-01
smoothness_SD	1.3e-01	5.0e-01	4.8e-01
compactness_SD	2.9e-02	4.1e-02	8.4e-01
concavity_SD	-1.1e-01	5.3e-01	4.7e-01
‘concave points_SD‘	-3.9e-02	3.3e-02	8.6e-01
symmetry_SD	9.3e-02	2.5e-01	6.1e-01
‘fractal dimension_SD‘	5.5e-02	1.4e-01	7.1e-01
radius_worst	-5.2e-02	1.6e-01	6.9e-01
texture_worst	1.1e-01	5.1e-01	4.7e-01
perimeter_worst	3.7e-02	1.1e-01	7.5e-01
area_worst	2.1e-02	1.7e-02	9.0e-01
smoothness_worst	-3.6e-02	7.3e-02	7.9e-01
compactness_worst	1.7e-01	1.8e+00	1.9e-01
concavity_worst	9.6e-02	4.2e-01	5.2e-01
‘concave points_worst‘	-1.3e-01	6.8e-01	4.1e-01
symmetry_worst	-2.1e-01	1.6e+00	2.0e-01
‘fractal dimension_worst‘	-1.1e-01	4.6e-01	5.0e-01
‘Tumor size‘	8.7e-02	3.3e-01	5.6e-01
‘Lymph node status‘	-2.0e-03	2.1e-04	9.9e-01
GLOBAL	NA	4.0e+01	1.5e-01

Aucune des p.valeur n’est inferieure à 0.05. Etanr donné que l’hypothèse nulle de ce test est la non-dependance en temps des résidus de schonfeld et compte tenu des p.valeur obtenues on peut conclure à la vérification de l’hypothèse de proportionnalité des risques relatifs (c’est à dire on ne rejette pas H_0).

On peut afficher les 4 premiers graphes de résidus:

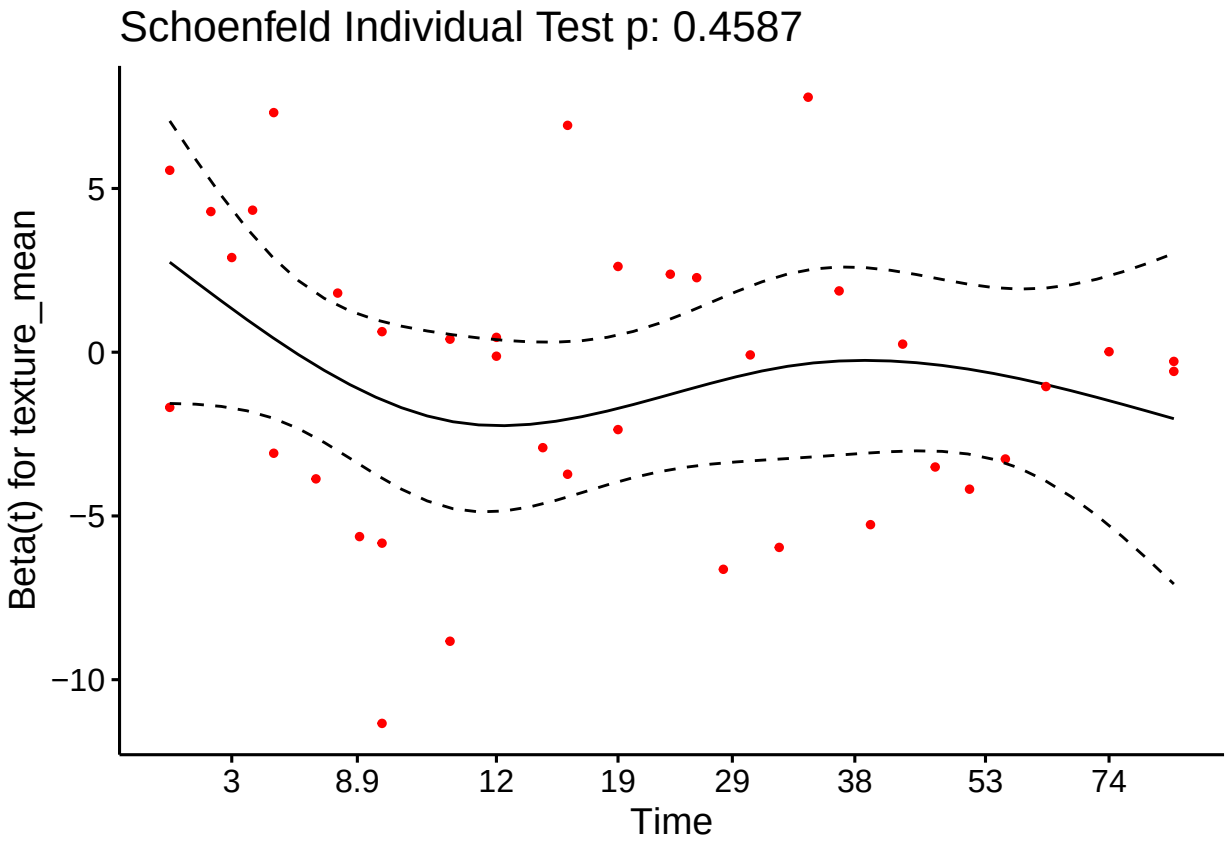
```
ggcoxzph(test.hyp.cox.all)[1]
```

```
## $`1`
```



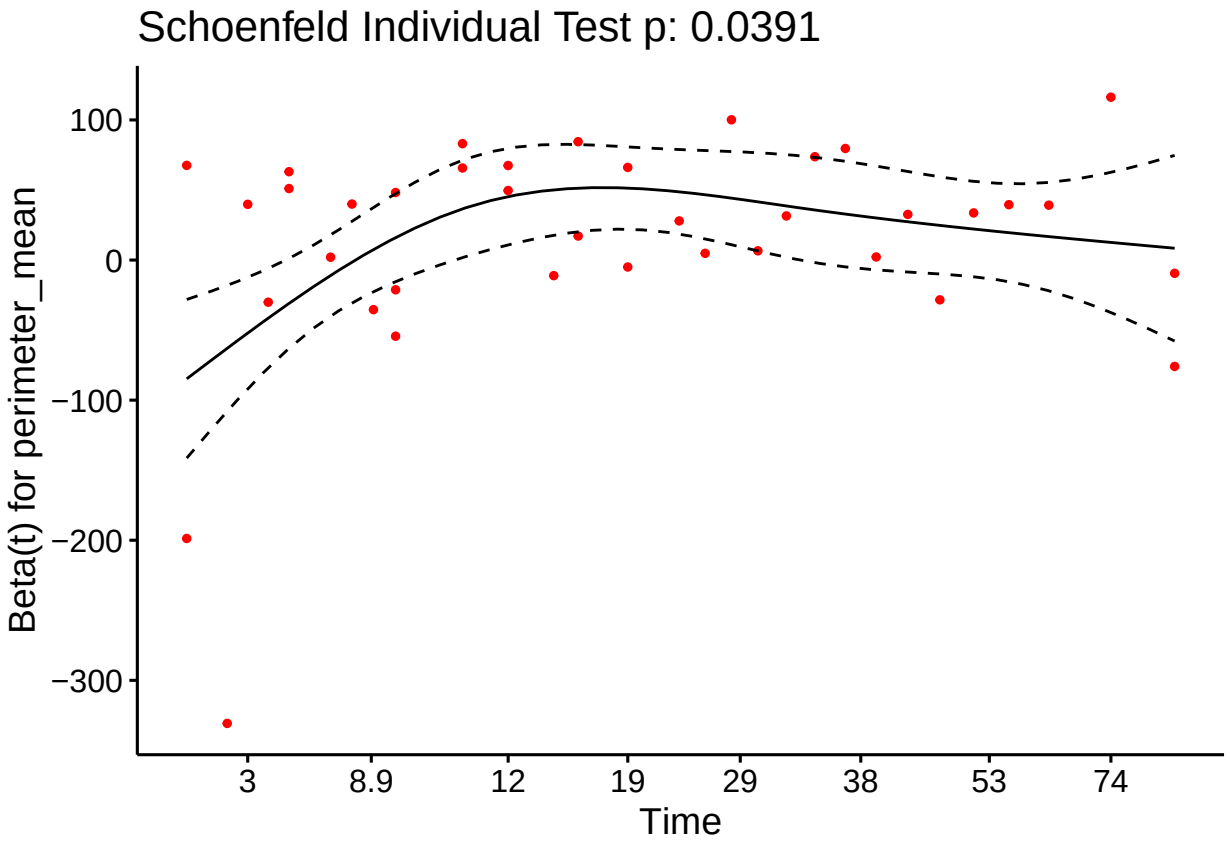

```
ggcoxzph(test.hyp.cox.all)[2]
```

```
## $`2`
```



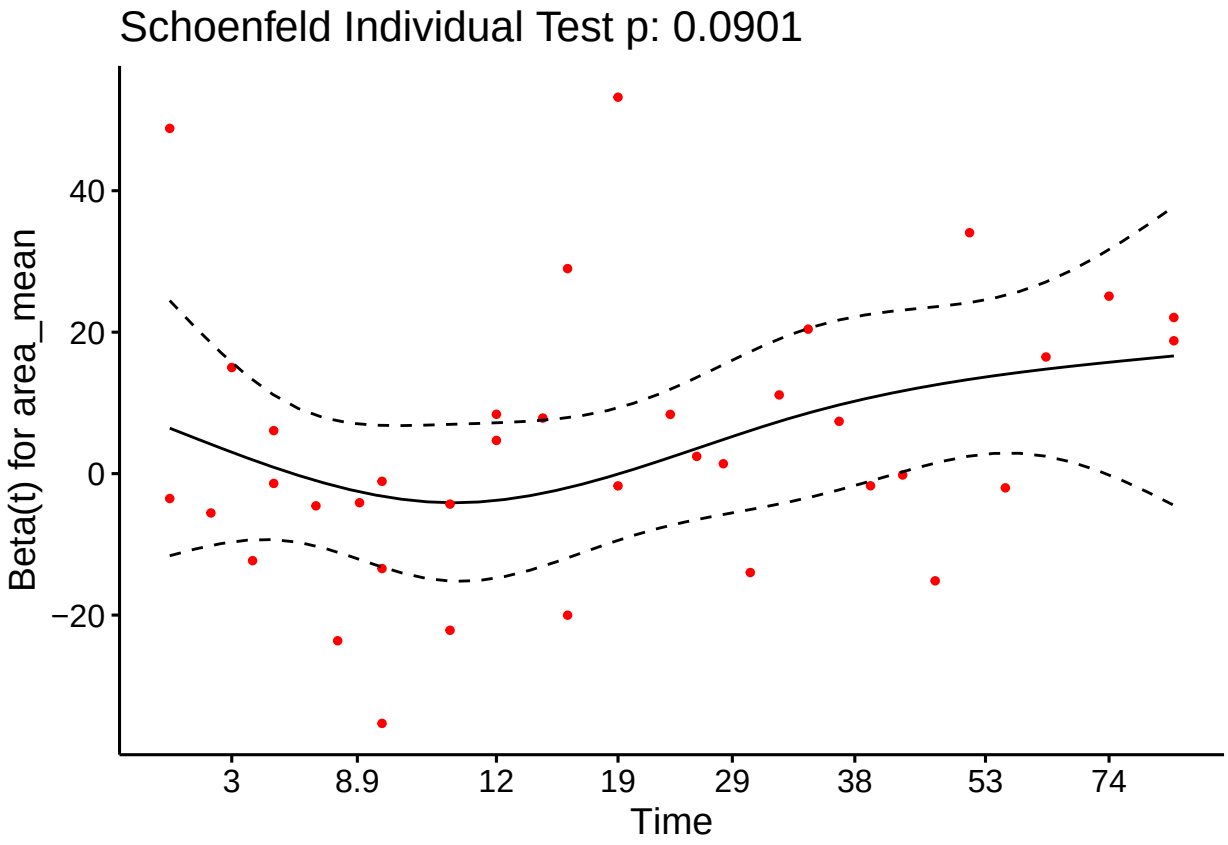

```
ggcoxzph(test.hyp.cox.all)[3]
```

```
## $`3`
```




```
ggcoxzph(test.hyp.cox.all)[4]
```

```
## $`4`
```



3.3 Step AIC

3.3.1 Cox model

On effectue alors une sélection de variables basée sur le critère AIC:

```
cox.AIC = stepAIC(cox.all,trace=F)
df.cox.AIC = data.frame(format(summary(cox.AIC)$conf.int[,1],scientific = T, digit = 2),
                        paste("[",format(summary(cox.AIC)$conf.int[,3], scientific = T, digit = 2),
                              ";",format(summary(cox.AIC)$conf.int[,4],scientific = T, digit = 2),"]"),
                        format(summary(cox.AIC)$coefficient[,5], scientific = T, digit = 2),
                        ifelse(summary(cox.AIC)$coefficient[,5] <= 0.05, "*","")
)

colnames(df.cox.AIC) = c("exp(coef)", "IC", "p", "Significativité ")
kable(df.cox.AIC, caption = "Modèle de cox avec toutes les covariables") %>%
  kable_styling(latex_options = "striped", stripe_color = "gray!6")
```

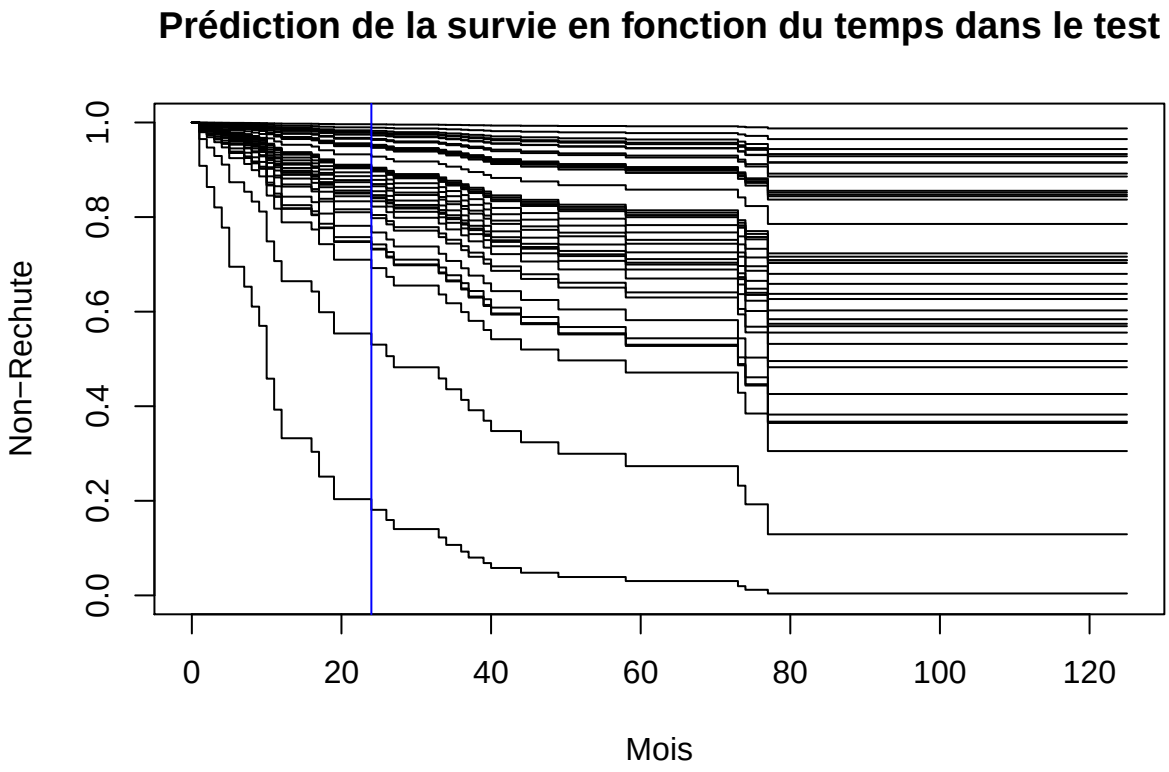
Le modèle final a retenu 12 variables.

3.3.2 Prediction, ROC et AUC

```
plot(survfit(cox.AIC , newdata = test.surv), xlab = "Mois", ylab="Non-Rechute", main = "Prédiction de la
abline(v = 24, col = "blue")
```

Table 7: Modèle de cox avec toutes les covariables

	exp(coef)	IC	p	Significativité
radius_mean	4.9e-07	[9.1e-11 ; 2.6e-03]	9.1e-04	*
perimeter_mean	5.2e+06	[5.9e+02 ; 4.6e+10]	8.4e-04	*
smoothness_mean	4.5e+00	[1.7e+00 ; 1.2e+01]	2.3e-03	*
concavity_mean	1.0e-01	[1.6e-02 ; 6.3e-01]	1.4e-02	*
'fractal dimension_mean'	2.5e-01	[1.0e-01 ; 6.0e-01]	2.1e-03	*
texture_SD	5.6e-01	[3.2e-01 ; 9.7e-01]	4.0e-02	*
perimeter_SD	1.7e+00	[1.0e+00 ; 2.8e+00]	3.7e-02	*
compactness_SD	3.9e+00	[1.3e+00 ; 1.2e+01]	1.6e-02	*
concavity_SD	3.1e-01	[1.0e-01 ; 9.3e-01]	3.6e-02	*
compactness_worst	2.1e-01	[6.3e-02 ; 6.8e-01]	9.8e-03	*
concavity_worst	7.1e+00	[2.0e+00 ; 2.5e+01]	2.3e-03	*
'Lymph node status'	1.5e+00	[1.1e+00 ; 2.0e+00]	5.7e-03	*




```
pred.cox.AIC = survfit(cox.AIC , newdata = test.surv)
prob.pred.24.AIC = pred.cox.AIC$surv[24,]
```

```
pred.24.AIC = ifelse(prob.pred.24.AIC <= 0.5, 1, 0)
pred.24.AIC = as.factor(pred.24.AIC)
```

On obtient alors la matrice de confusion suivante (Table 8):

Table 8: Matrice de confusion pour la prédiction dans le modele cox.AIC

	Real.0	Real.1
0	29	8
1	1	1

```
kable(table(Predicted = pred.24.AIC, Real = test.surv$recurrent),
      caption = "Matrice de confusion pour la prédiction dans le modele cox.AIC",
      col.names = c("Real.0","Real.1"))
```

Le modèle a du mal à prédire correctement les cas de rechute. Cela est peut-être du au faible nombre de rechutes dans le jeu de données par rapport à celui de non-rechutes.

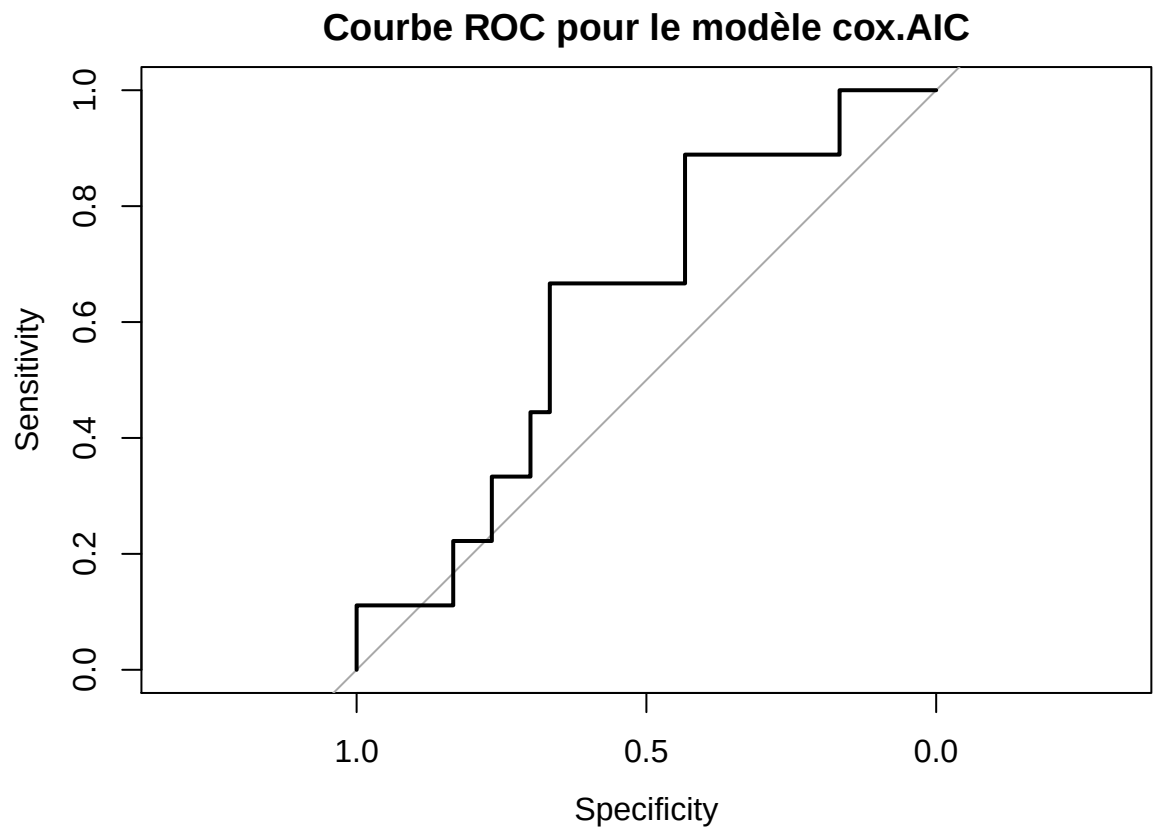
En faisant varier le seuil entre 0 et 1 on obtient la courbe ROC suivante:

```
roc.cox.AIC = roc(response = test.surv$recurrent, predictor = prob.pred.24.AIC)
```

```
## Setting levels: control = FALSE, case = TRUE
```

```
## Setting direction: controls > cases
```

```
plot(roc.cox.AIC, main = "Courbe ROC pour le modèle cox.AIC")
```




```
auc.cox.AIC = auc(roc.cox.AIC)
cat("L'AUC du modèle cox.AIC est de:",auc.cox.AIC)
```

```
## L'AUC du modèle cox.AIC est de: 0.6296296
```

De même que précédement, on fixe un seuil à 0.5 et on calcule la précision du cox.AIC:

```
pred.24.AIC.bis = ifelse(prob.pred.24.AIC <= 0.5, 1, 0)
pred.24.AIC.bis = as.factor(pred.24.AIC.bis)
table(Pred = pred.24.AIC.bis, real = test.surv$recurrent)
```

```
##     real
## Pred FALSE TRUE
##    0    29    8
##    1     1    1
```

On a donc une précision de $ (29+1)/39 = 0.7692308$.

3.3.3 Diagnostique du modèle

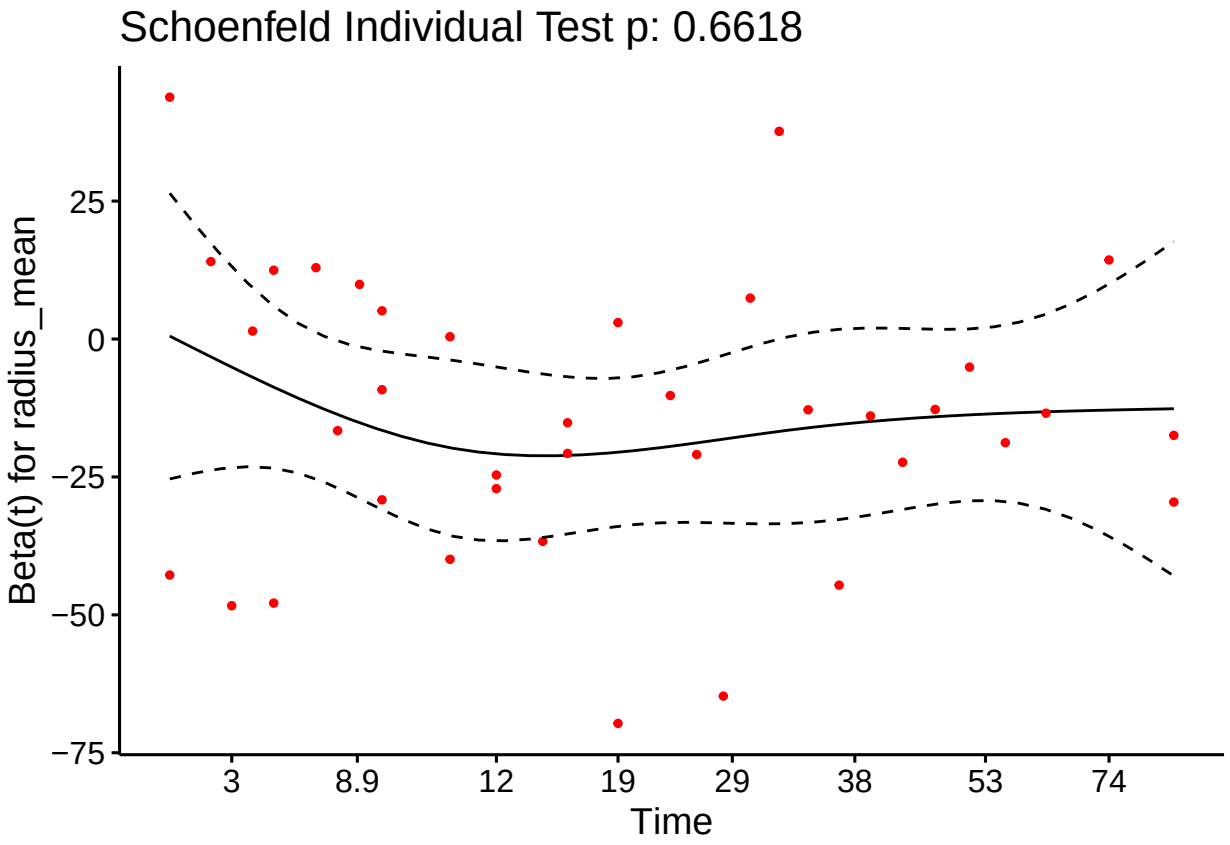
```
test.hyp.cox.AIC <- cox.zph(cox.AIC)
kable(format(as.matrix(test.hyp.cox.AIC$table), scientific = T, digit = 2)) %>%
  kable_styling(latex_options = "striped", stripe_color = "gray!6")
```

	rho	chisq	p
radius_mean	-7.8e-02	1.9e-01	6.6e-01
perimeter_mean	7.2e-02	1.8e-01	6.8e-01
smoothness_mean	7.7e-02	2.7e-01	6.1e-01
concavity_mean	-1.4e-01	7.8e-01	3.8e-01
‘fractal dimension_mean‘	-2.3e-02	2.0e-02	8.9e-01
texture_SD	2.2e-01	2.0e+00	1.6e-01
perimeter_SD	7.3e-02	2.2e-01	6.4e-01
compactness_SD	4.0e-02	6.7e-02	8.0e-01
concavity_SD	3.9e-02	6.4e-02	8.0e-01
compactness_worst	-2.0e-02	1.5e-02	9.0e-01
concavity_worst	1.1e-01	4.3e-01	5.1e-01
‘Lymph node status‘	1.1e-01	4.0e-01	5.3e-01
GLOBAL	NA	1.1e+01	5.7e-01

Aucune des p.valeur n’est inférieure à 0.05. Etant donné que l’hypothèse nulle de ce test est la non-dépendance en temps des résidus de Schonfeld et, compte tenu des p.valeur obtenues, on peut conclure à la vérification de l’hypothèse de proportionnalité des risques relatifs (c’est à dire on ne rejette pas H_0).

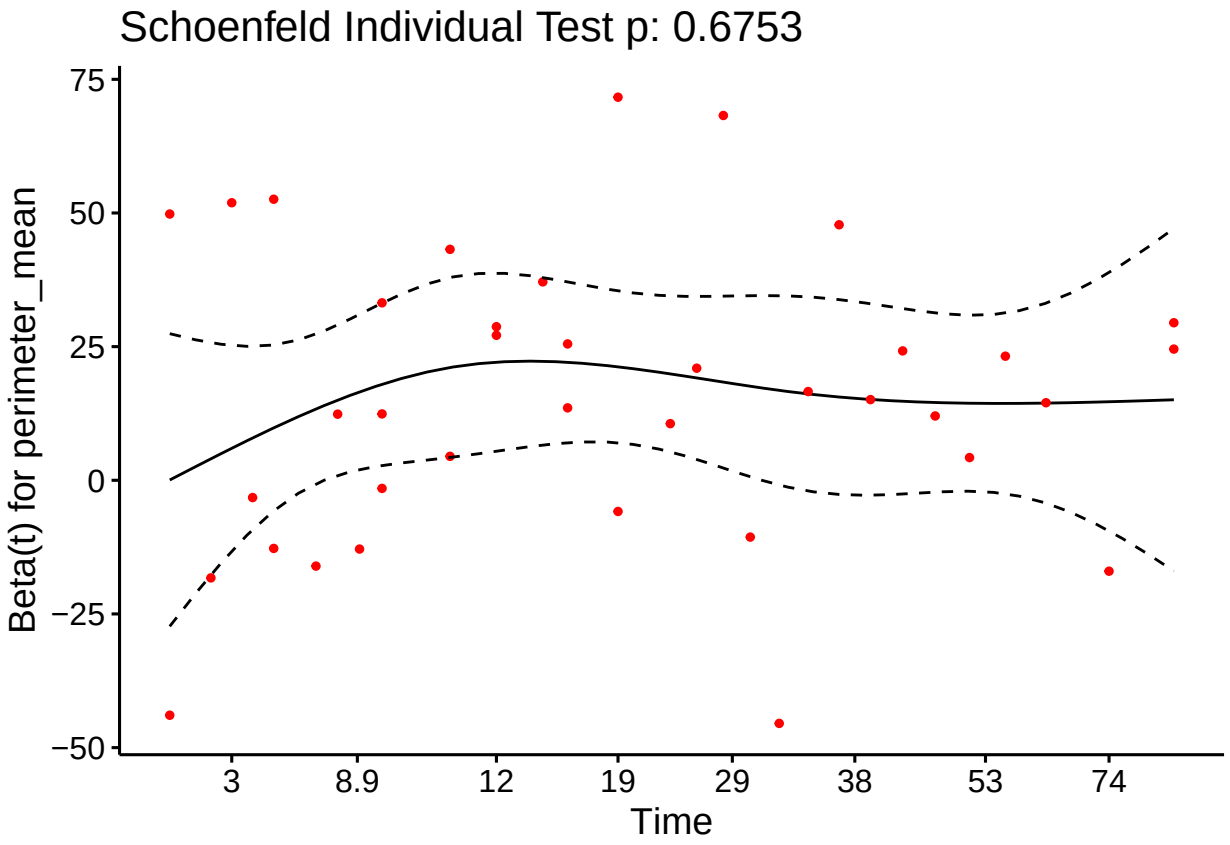
```
ggcoxzph(test.hyp.cox.AIC)[1]
```

```
## $`1`
```



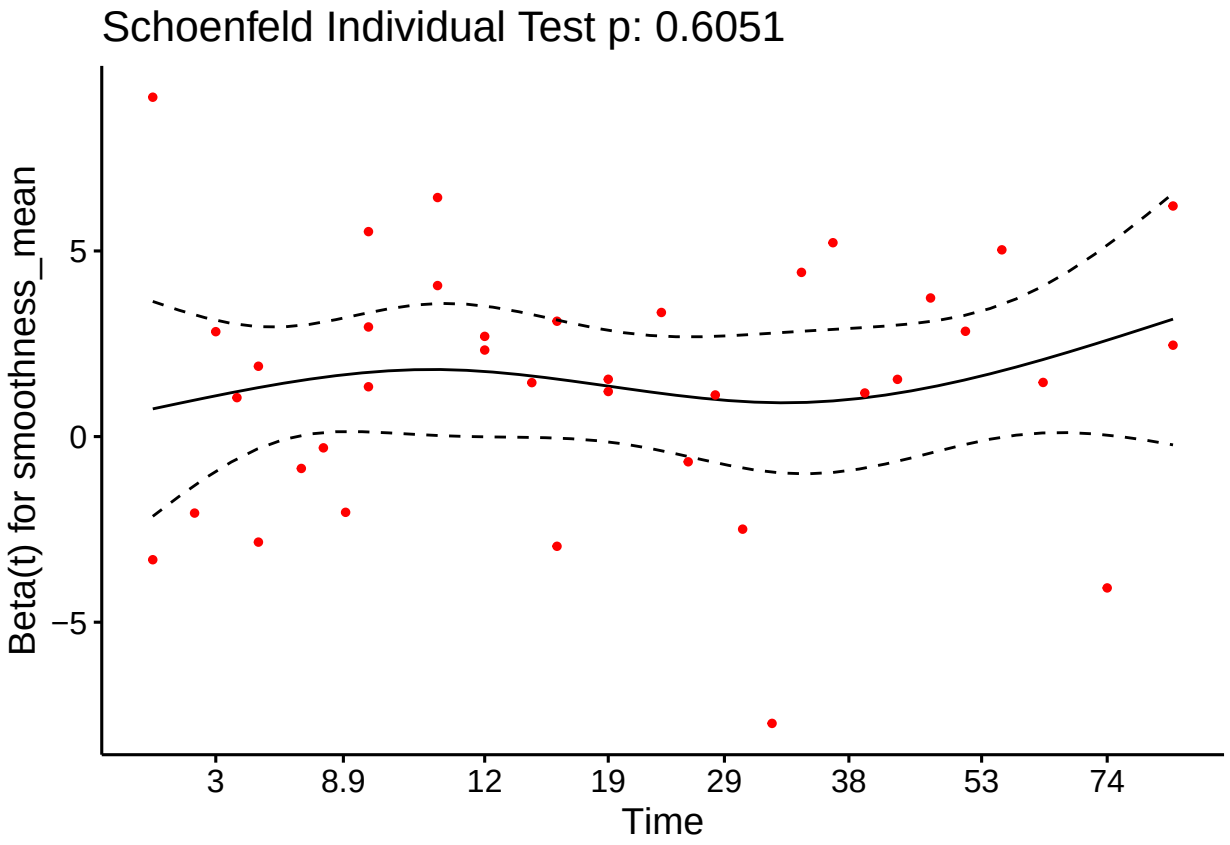

```
ggcoxzph(test.hyp.cox.AIC)[2]
```

```
## $`2`
```



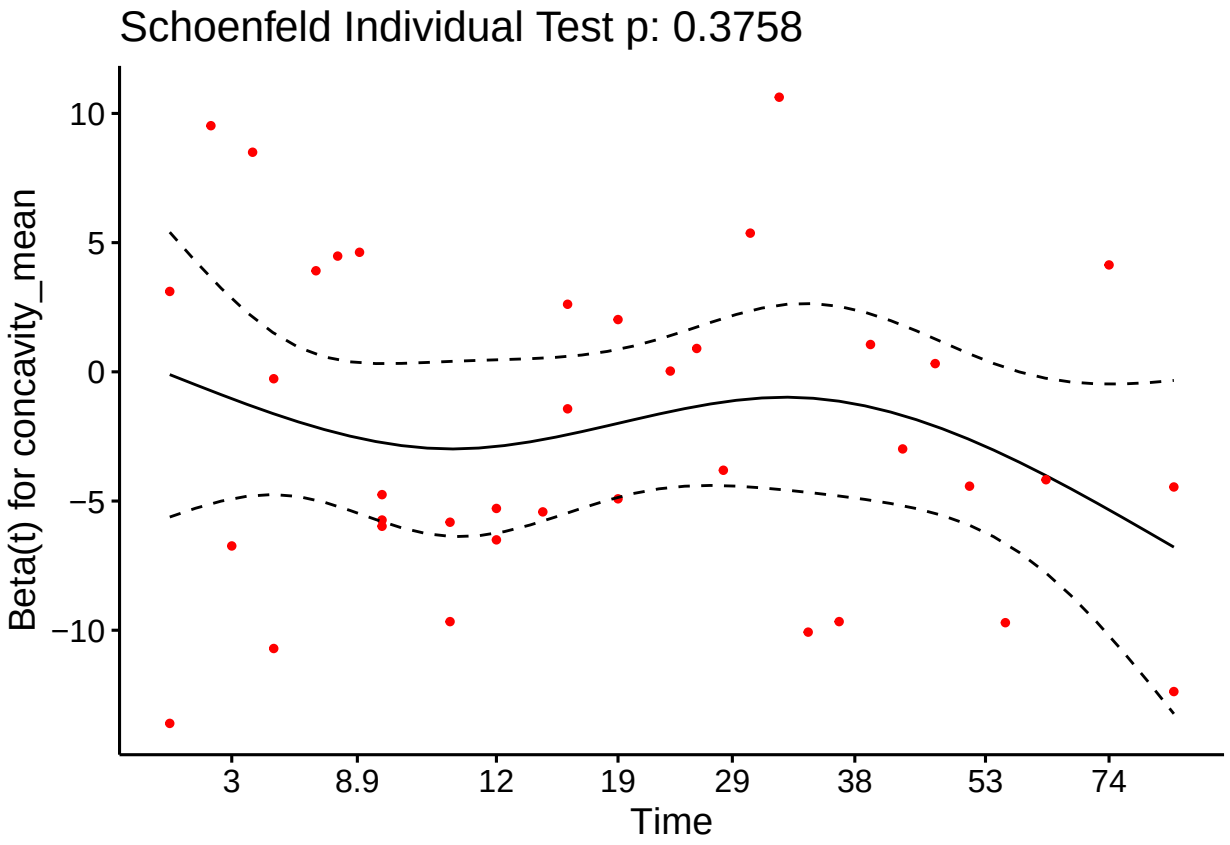

```
ggcoxzph(test.hyp.cox.AIC)[3]
```

```
## $`3`
```




```
ggcoxzph(test.hyp.cox.AIC)[4]
```

```
## $`4`
```



4. Classification

Afin de comparer les résultats des modèles de classification avec ceux du modèle `cox.AIC` et `cox.all`, pour chaque classifieur nous feraont deux prédition: l'une en se basant sur les variables du modèle complet et l'autre sur les varaible du modèle AIC.

4.1 Création du train et du test

4.1.1 Avec toutes les variables

```
var.all = c( "recurrent","radius_mean","texture_mean","perimeter_mean","area_mean",
             "smoothness_mean","compactness_mean" ,"concavity_mean",
             "concave points_mean","symmetry_mean", "fractal dimension_mean",
             "radius_SD","texture_SD","perimeter_SD","area_SD"            ,"smoothness_SD"          ,
 ,"concavity_SD"           ,   "concave points_SD"
 ,"symmetry_SD"            ,   "fractal dimension_SD"
 ,"radius_worst"           ,   "texture_worst"
 ,"perimeter_worst"        ,   "area_worst"
 ,"smoothness_worst"       ,   "compactness_worst"
 ,"concavity_worst"        ,   "concave points_worst"
 ,"symmetry_worst"         ,   "fractal dimension_worst"
 ,"Tumor size"           ,   "Lymph node status"    )
```

```
train.clf.all = train[, var.all]
test.clf.all = test[, var.all]
```

4.1.2 Avec les variables issues de l'AIC

```
var.AIC = c("recurrent", "radius_mean", "perimeter_mean", "smoothness_mean", "concavity_mean", "fractal
    "texture_SD", "perimeter_SD", "compactness_SD", "concavity_SD", "compactness_worst", "concavity_wor
train.clf.AIC = train[, var.AIC]
test.clf.AIC = test[, var.AIC]
```

4.2 Prédictions, ROC, AUC et Accuracy sur les variables issues de l'AIC

4.2.1 Naives Bayes

```
mod.NB.AIC = naiveBayes(recurrent ~ ., data = train.clf.AIC)
pred.NB.AIC = predict(mod.NB.AIC, subset(test.clf.AIC, select = -recurrent))
cm.NB.AIC = confusionMatrix(as.factor(pred.NB.AIC), as.factor(test.clf.AIC$recurrent))
cm.NB.AIC$table
```

```
##            Reference
## Prediction FALSE TRUE
##      FALSE    28    7
##      TRUE      2    2
```

```
acc.NB.AIC = cm.NB.AIC$overall[1]
acc.NB.AIC
```

```
##  Accuracy
## 0.7692308
```

Malgré une précision correcte sur le jeu de données test, la matrice de confusion nous informe que la majorité des erreurs de classification commises par l'algorithme sont dues à une mauvaise prédiction des cas de rechute (`reccurent`=TRUE). Seulement ce sont bien ces cas que l'on cherche à prédire de la meilleure manière possible.

4.2.2 Linear Discriminant Analysis

```
mod.lda.AIC = lda(recurrent ~ ., data = train.clf.AIC)
pred.lda.AIC = predict(mod.lda.AIC, subset(test.clf.AIC, select = -recurrent))
cm.lda.AIC = confusionMatrix(as.factor(pred.lda.AIC$class), as.factor(test.clf.AIC$recurrent))
cm.lda.AIC$table
```

```
##            Reference
## Prediction FALSE TRUE
##      FALSE    29    7
##      TRUE      1    2
```

```
acc.lda.AIC = cm.lda.AIC$overall[1]
acc.lda.AIC
```

```
##  Accuracy
## 0.7948718
```

Résultats similaires à la méthode précédente, même erreur de prédiction.

4.2.3 Quadratic Discriminant Analysis

```
mod.qda.AIC = qda(recurrent ~ ., data=train.clf.AIC)
pred.qda.AIC = predict(mod.qda.AIC, subset(test.clf.AIC, select = -recurrent))
cm.qda.AIC = confusionMatrix(as.factor(pred.qda.AIC$class), as.factor(test.clf.AIC$recurrent))
cm.qda.AIC$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    25    7
##      TRUE      5    2
```

```
acc.qda.AIC = cm.qda.AIC$overall[1]
acc.qda.AIC
```

```
##  Accuracy
## 0.6923077
```

Résultats semblables, même moins précis.

4.2.4. Support Vector Machine

```
mod.svm.AIC <- svm(recurrent ~ ., data=train.clf.AIC, type="C")
pred.svm.AIC = predict(mod.svm.AIC, subset(test.clf.AIC, select = -recurrent))
cm.svm.AIC = confusionMatrix(as.factor(pred.svm.AIC), as.factor(test.clf.AIC$recurrent))
cm.svm.AIC$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    30    9
##      TRUE      0    0
```

```
acc.svm.AIC = cm.svm.AIC$overall[1]
acc.svm.AIC
```

```
##  Accuracy
## 0.7692308
```

Résultats encore moins bons.

4.2.5. kNN

```
mod.knn.AIC = as.factor(knn(train=train.clf.AIC[-c(1)], test=test.clf.AIC[-c(1)], cl=train.clf.AIC$recu
cm.knn.AIC = confusionMatrix(as.factor(mod.knn.AIC), as.factor(test.clf.AIC$recurrent))
cm.knn.AIC$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    29    9
##      TRUE      1    0
```

```
acc.knn.AIC = cm.knn.AIC$overall[1]
acc.knn.AIC
```

```
##  Accuracy
## 0.7435897
```

Résultats similaires. ### 4.2.6 Random Forest

```
train.recurrent.AIC = as.factor(train.clf.AIC$recurrent)
test.recurrent.AIC = as.factor(test.clf.AIC$recurrent)
mod.rf.AIC = randomForest(train.clf.AIC[,-c(1)], train.recurrent.AIC)
pred.rf.AIC = predict(mod.rf.AIC, newdata=test.clf.AIC[,-c(1)], type="class")
cm.rf.AIC = confusionMatrix(pred.rf.AIC, as.factor(test.recurrent.AIC))
cm.rf.AIC$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    30    8
##      TRUE      0    1
```

```
acc.rf.AIC = cm.rf.AIC$overall[1]
acc.rf.AIC
```

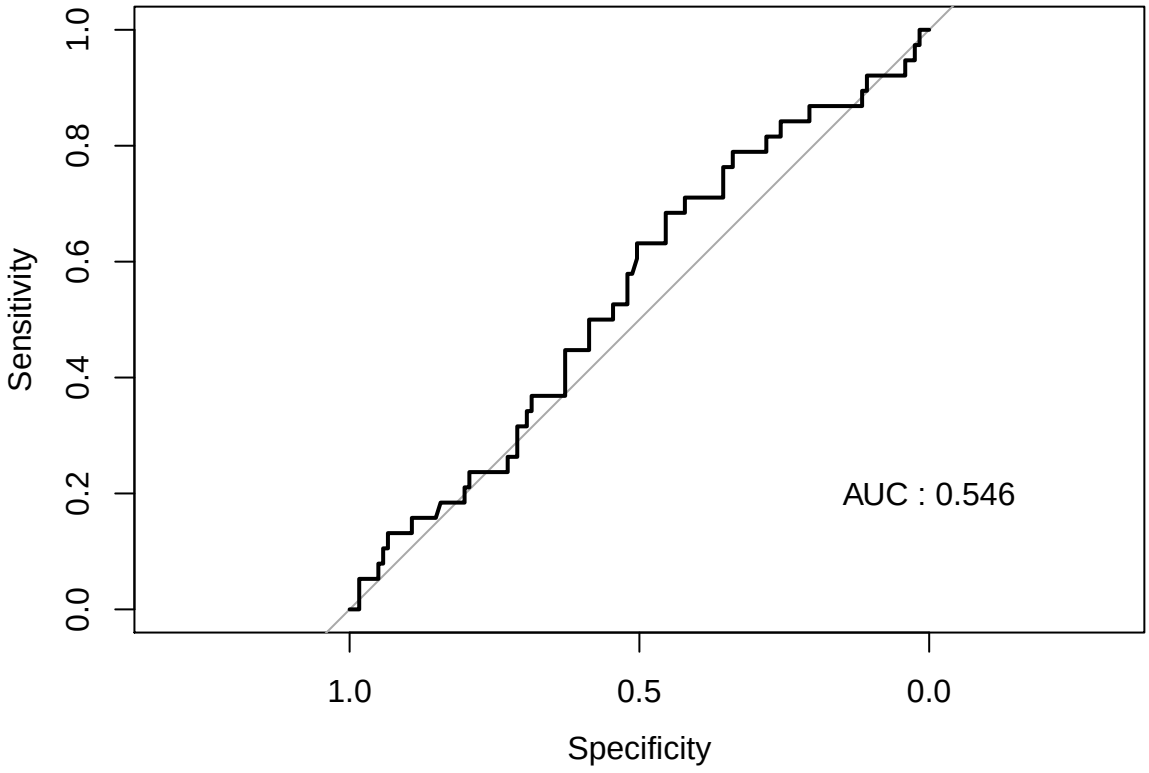
```
##  Accuracy
## 0.7948718
```

```
roc.rf.AIC = roc(train.clf.AIC$recurrent, mod.rf.AIC$votes[,2] )
```

```
## Setting levels: control = FALSE, case = TRUE
```

```
## Setting direction: controls < cases
```

```
plot(roc.rf.AIC)
auc.rf.AIC = paste("AUC :", round(auc(roc.rf.AIC),3))
text(0.0, 0.2, auc.rf.AIC)
```



4.3. Prédictions, ROC, AUC et Accuracy sur les variables issues du modèle complet

4.3.1 Naives Bayes

```
mod.NB.all = naiveBayes(recurrent ~ ., data = train.clf.all)
pred.NB.all = predict(mod.NB.all, subset(test.clf.all, select = -recurrent))
cm.NB.all = confusionMatrix(as.factor(pred.NB.all), as.factor(test.clf.all$recurrent))
cm.NB.all$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    25    7
##      TRUE      5    2
```

```
acc.NB.all = cm.NB.all$overall[1]
acc.NB.all
```

```
##  Accuracy
## 0.6923077
```

Malgré une précision correcte sur le jeu de données test, la matrice de confusion nous informe que la majorité des erreurs de classification commises par l'algorithme sont dues à une mauvaise prédiction des cas de rechute (`reccurent`=TRUE). Seulement ce sont bien ces cas que l'on cherche à prédire de la meilleure manière possible.

4.3.2 Linear Discriminant Analysis

```
mod.lda.all = lda(recurrent ~ ., data = train.clf.all)
pred.lda.all = predict(mod.lda.all, subset(test.clf.all, select = -recurrent))
cm.lda.all = confusionMatrix(as.factor(pred.lda.all$class), as.factor(test.clf.all$recurrent))
cm.lda.all$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    26    8
##      TRUE      4    1
```

```
acc.lda.all = cm.lda.all$overall[1]
acc.lda.all
```

```
##  Accuracy
## 0.6923077
```

Résultats similaires à la méthode précédente, même erreur de prédiction.

4.3.3 Quadratic Discriminant Analysis

```
mod.qda.all = qda(recurrent ~ ., data=train.clf.all)
pred.qda.all = predict(mod.qda.all, subset(test.clf.all, select = -recurrent))
cm.qda.all = confusionMatrix(as.factor(pred.qda.all$class), as.factor(test.clf.all$recurrent))
cm.qda.all$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    30    9
##      TRUE      0    0
```

```
acc.qda.all = cm.qda.all$overall[1]
acc.qda.all
```

```
##  Accuracy
## 0.7692308
```

Résultats semblables, même moins précis.

4.3.4. Support Vector Machine

```
mod.svm.all <- svm(recurrent ~ ., data=train.clf.all, type="C")
pred.svm.all = predict(mod.svm.all, subset(test.clf.all, select = -recurrent))
cm.svm.all = confusionMatrix(as.factor(pred.svm.all), as.factor(test.clf.all$recurrent))
cm.svm.all$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    30    9
##      TRUE      0    0
```

```
acc.svm.all = cm.svm.all$overall[1]
acc.svm.all
```

```
##  Accuracy
## 0.7692308
```

Résultats encore moins bons.

4.3.5. kNN

```
mod.knn.all = as.factor(knn(train=train.clf.all[-c(1)], test=test.clf.all[-c(1)], cl=train.clf.all$recu
cm.knn.all = confusionMatrix(as.factor(mod.knn.all), as.factor(test.clf.all$recurrent))
cm.knn.all$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    28    9
##      TRUE      2    0
```

```
acc.knn.all = cm.knn.all$overall[1]
acc.knn.all
```

```
##  Accuracy
## 0.7179487
```

Résultats similaires.

4.3.6. Random Forest

```
train.recurrent.all = as.factor(train.clf.all$recurrent)
test.recurrent.all = as.factor(test.clf.all$recurrent)
mod.rf.all = randomForest(train.clf.all[,-c(1)], train.recurrent.all)
pred.rf.all = predict(mod.rf.all, newdata=test.clf.all[,-c(1)], type="class")
cm.rf.all = confusionMatrix(pred.rf.all, as.factor(test.recurrent.all))
cm.rf.all$table
```

```
##           Reference
## Prediction FALSE TRUE
##      FALSE    30    9
##      TRUE      0    0
```

```
acc.rf.all = cm.rf.all$overall[1]
acc.rf.all
```

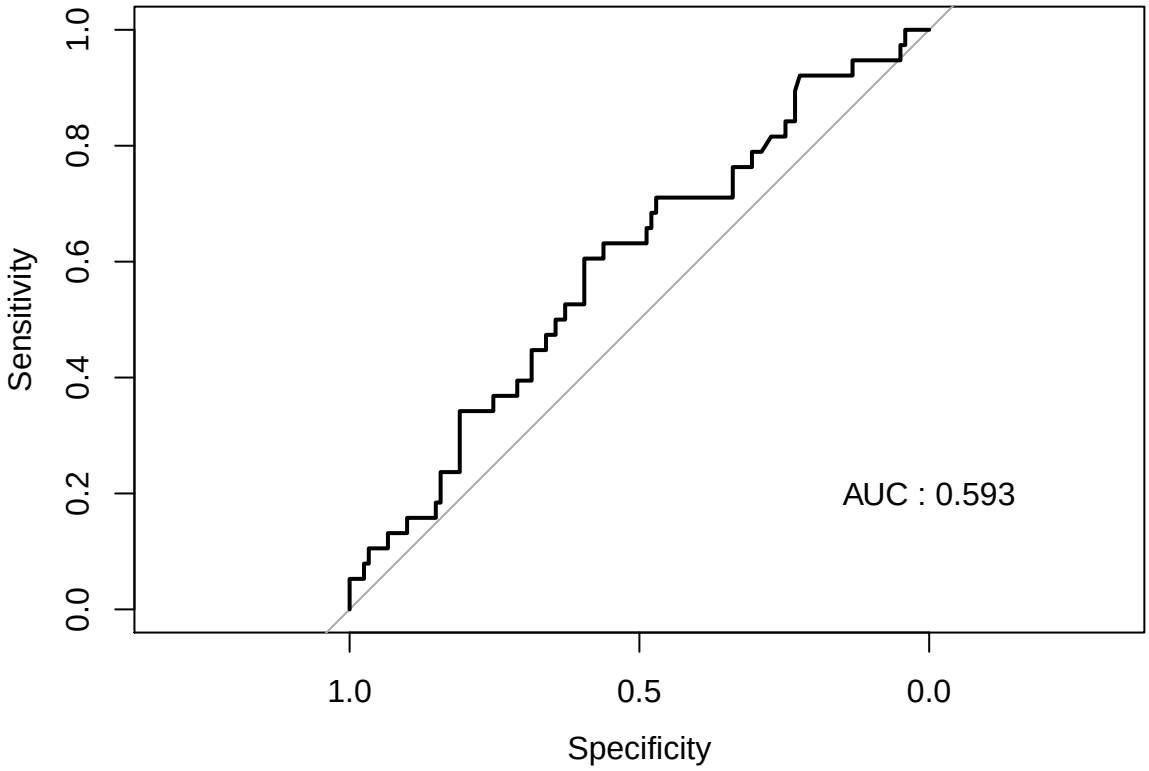
```
##  Accuracy
## 0.7692308
```

```
roc.rf.all = roc(train.clf.all$recurrent, mod.rf.all$votes[,2] )
```

```
## Setting levels: control = FALSE, case = TRUE
```

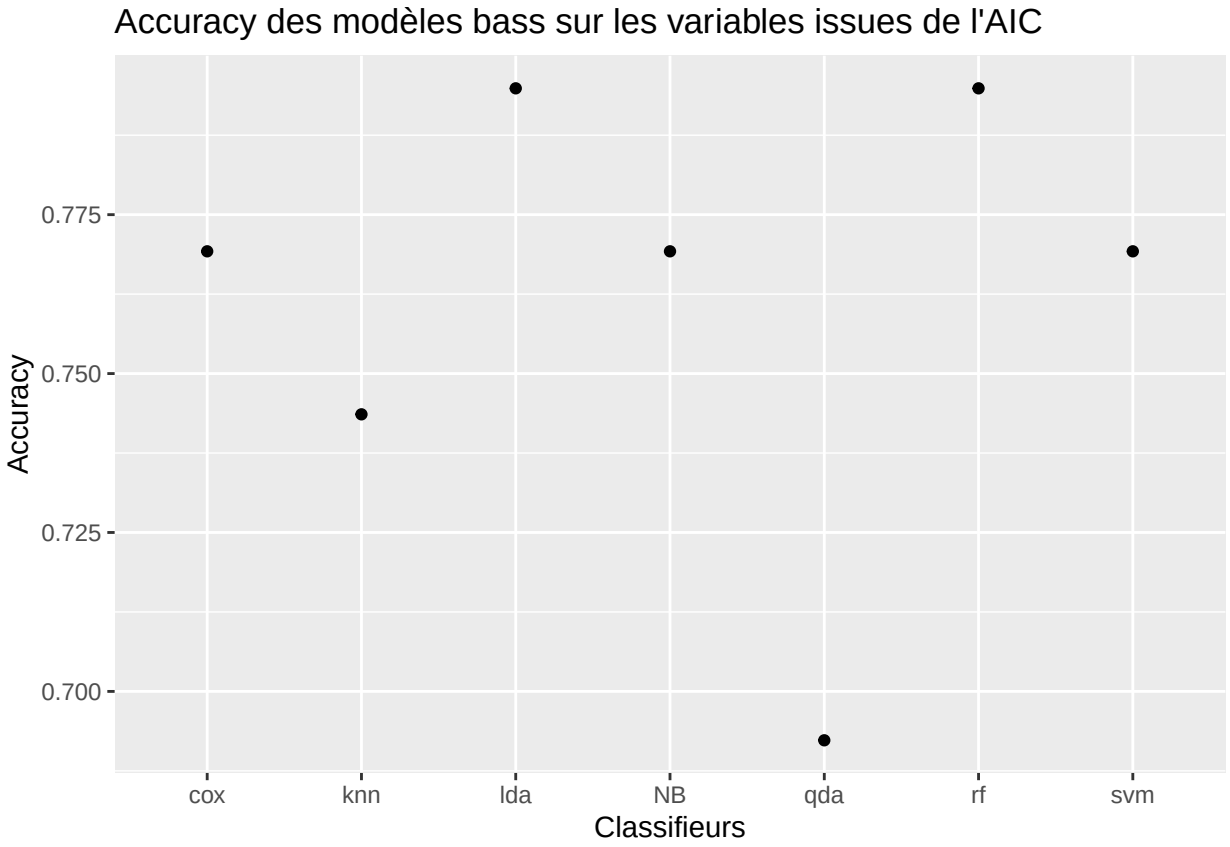
```
## Setting direction: controls < cases
```

```
plot(roc.rf.all)
auc.rf.all = paste("AUC :", round(auc(roc.rf.all),3))
text(0.0, 0.2, auc.rf.all)
```



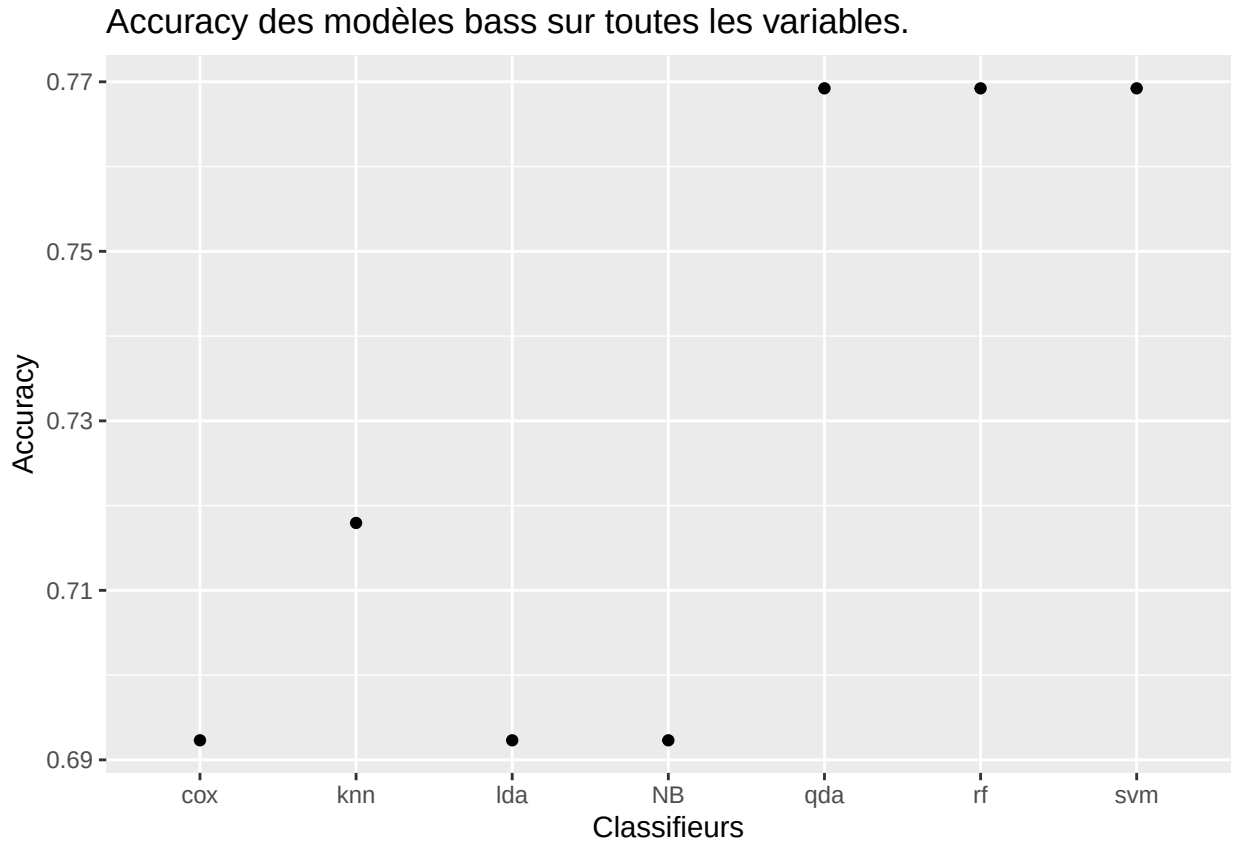
5. Résumé des performances des modèles

```
clf = c("knn","lda","NB","qda","rf","svm","cox")
acc.AIC = c(acc.knn.AIC, acc.lda.AIC, acc.NB.AIC, acc.qda.AIC,
            acc.rf.AIC, acc.svm.AIC, acc.cox.AIC)
df.acc.AIC = data.frame(Classifieurs = clf, Accuracy = acc.AIC)
ggplot(df.acc.AIC, aes(x = Classifieurs,y = Accuracy)) + geom_point() + ggtitle("Accuracy des modèles ba
```




```
#auc.AIC = c(auc.cox.AIC, auc.rf.AIC)
```

```
acc.all = c(acc.knn.all, acc.lda.all, acc.NB.all, acc.qda.all,
            acc.rf.all, acc.svm.all,acc.cox.all)
df.acc.all = data.frame(Classifieurs = clf, Accuracy = acc.all)
ggplot(df.acc.all, aes(x = Classifieurs,y = Accuracy)) + geom_point() + ggtitle("Accuracy des modèles ba
```




```
#auc.all = c(auc.cox.all, auc.rf.all)
```

6. Conclusion

Un problème récurrent, auxquels tous les algorithmes utilisés sont sensibles, s'observe dans ce contexte de classification: les cas de rechutes sont très mal prédits, car sous-représentés dans le jeu de données. Une manière de pallier cela serait d'équilibrer la répartition des classes dans les différents jeux de données. Seulement, nous ne disposons pas d'un nombre d'observationssuffisant pour lesquelles une rechute est observée à 24 mois.