Survival analysis or classification ?

Devoir maison obligatoire (Dauphine)

Paul Hardouin

31 Janvier 2020

Contents

Nous travaillons sur le jeu de données **wpbc**, disponible sur
https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wpbc.data.

Il est présenté sur
https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wpbc.names.

On souhaite prévoir la probabilité de rechute ("recurrent") à 24 mois. Pour cela, on veut comparer les méthodes de l'analyse de survie (modèles de Cox, survival random forests, ...) aux méthodes de classification. Les mesures de performances (notamment l'AUC) se feront sur un sous-échantillon de test forme de 20% a 30% des données, en faisant attention à bien stratifier.

1. Création du label pour la tâche de classification

On charge les librairies.

```
library(survival)
library(fitdistrplus)
library(tidyverse)
library(KMsurv)
library(ggfortify)
library(caret)
library(pROC)
```

On lit les données, on renomme les variables, et on transforme les variables **lymph** et **recur** en données numériques.

```
wpbc = read_csv("wpbc.data",col_names = F)
names(wpbc) = c("id","recur","time",paste0("V",c(1:30)),"tumor_size","lymph")
wpbc$lymph=as.numeric(wpbc$lymph)
wpbc$recur=as.numeric(wpbc$recur=="R")
```

On crée ensuite le label qui sera nécessaire pour la tâche de classification.

```
wpbc$label = wpbc$recur & wpbc$time<=24
wpbc$label=as.numeric(wpbc$label)
```

Quelques individus n'ont pas de valeur pour la variable **lymph**. Comme cette variable est pertinente pour les modèles de prédiction (comme nous le verrons par la suite), et que cela ne concerne que 4/198 individus, répartis à proportion vis-à-vis du label, nous faisons le choix de supprimer ces 4 individus du jeu de données.

```
wpbc = wpbc[!is.na(wpbc$lymph),]
```

2. Création des jeux TRAIN / TEST

On fixe la racine du générateur aléatoire. On crée ensuite un jeu de données test [30%] et un jeu train [70%], en faisant attention à bien stratifier selon le label de classification.

```
set.seed(123)
train.index <- createDataPartition(wpbc$label, p = .7, list = FALSE)
train <- wpbc[ train.index,]
test  <- wpbc[-train.index,]
```

3. Construction d'un modèle de Cox et d'un modèle de régression logistique

3.1. Construction d'un modèle de Cox

Pour ce modèle, on disqualifie d'entrée les covariables **id** et **label**.
On fait une recherche pas à pas en utilisant le critère AIC.

COX : on obtient un modèle avec 7 covariables.

```
cox_model = coxph(Surv(time,recur)~.-id-label, data = train)
cox_AIC = stepAIC(cox_model, trace = F)
cox_AIC
```

```
## Call:
## coxph(formula = Surv(time, recur) ~ V1 + V2 + V3 + V6 + V23 +
##     V25 + lymph, data = train)
##
##              coef  exp(coef)   se(coef)      z        p
## V1    -5.261e+00  5.190e-03  2.001e+00 -2.630 0.008547
## V2    -1.722e-01  8.418e-01  6.115e-02 -2.817 0.004854
## V3     7.560e-01  2.130e+00  3.003e-01  2.518 0.011808
## V6    -3.878e+01  1.432e-17  1.447e+01 -2.680 0.007354
## V23    6.041e-02  1.062e+00  1.731e-02  3.489 0.000485
## V25    2.473e+01  5.503e+10  1.368e+01  1.808 0.070582
## lymph  1.058e-01  1.112e+00  2.722e-02  3.887 0.000102
##
## Likelihood ratio test=36.93  on 7 df, p=4.845e-06
## n= 136, number of events= 28
```

3.2. Construction d'un modèle de régression logistique

Pour ce modèle, on disqualifie d'entrée les covariables **id**, **recur**, et **time**.
On fait une recherche pas à pas en utilisant le critère AIC (par defaut).
On fait une recherche avec la méthode progressive.

CLASSIFICATION : on obtient un modèle avec 5 covariables.

```
m1 = glm(label~.-id-recur-time, family=binomial, data=train)
m0 = glm(label~1,               family=binomial, data=train)
glm_AIC = step(m0, direction="both", scope=list(upper=m1,lower=m0))
summary(glm_AIC)
```

```
##
## Call:
## glm(formula = label ~ V24 + lymph + V9 + V25 + V2, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4808  -0.4391  -0.2627  -0.1298   2.9027
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.785e-01  3.773e+00  -0.233 0.815874
## V24          1.925e-03  5.568e-04   3.458 0.000544 ***
## lymph        1.096e-01  4.318e-02   2.537 0.011172 *
## V9          -4.495e+01  1.913e+01  -2.349 0.018808 *
## V25          4.605e+01  1.951e+01   2.360 0.018288 *
## V2          -1.363e-01  8.570e-02  -1.590 0.111852
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 102.481  on 135  degrees of freedom
## Residual deviance:  69.963  on 130  degrees of freedom
## AIC: 81.963
##
## Number of Fisher Scoring iterations: 6
```

4. Prédiction des probabilités de rechute à 24 mois et matrices de confusion

4.1. Prédicion avec le modèle de Cox

Pour faire notre prédiction de rechute, on cherche d'abord à prédire la probabilité de survie (non rechute) à 24 mois. On prend alors la probabilité complémentaire pour obtenir la probabilité de rechute à 24 mois.

```
# Modèle de prédiction
  cox_final = coxph(cox_AIC$call$formula, data = train)
  prediction_model = survfit(cox_final)
  marqueurs = predict(cox_final,test)
  time = prediction_model$time
# Prédiction : on fait 1-.. car on veut pr
  cox_prediction = 1-exp(-prediction_model$cumhaz[time==24]*exp(marqueurs))
  cox_classification = round(cox_prediction)
# Matrice de confusion
  confusionMatrix(factor(cox_classification),factor(test$label))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 44  9
##          1  3  2
##
##                Accuracy : 0.7931
##                  95% CI : (0.6665, 0.8883)
##     No Information Rate : 0.8103
##     P-Value [Acc > NIR] : 0.7012
##
##                   Kappa : 0.1491
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.9362
##             Specificity : 0.1818
##          Pos Pred Value : 0.8302
##          Neg Pred Value : 0.4000
##              Prevalence : 0.8103
##          Detection Rate : 0.7586
##    Detection Prevalence : 0.9138
##       Balanced Accuracy : 0.5590
##
##        'Positive' Class : 0
##
```

4.2. Prédiction avec le modèle logistique

```
# Modèle de prédiction
   glm_final = glm(glm_AIC$call$formula, family = binomial, data = train)
# Prédiction
  glm_prediction = predict(glm_final, test, type = "response")
  glm_classification = round(glm_prediction)
# Matrice de confusion
  confusionMatrix(factor(glm_classification),factor(test$label))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 46  9
##          1  1  2
##
##                Accuracy : 0.8276
##                  95% CI : (0.7057, 0.9141)
##     No Information Rate : 0.8103
##     P-Value [Acc > NIR] : 0.44714
##
##                   Kappa : 0.2225
##
##  Mcnemar's Test P-Value : 0.02686
##
##             Sensitivity : 0.9787
##             Specificity : 0.1818
##          Pos Pred Value : 0.8364
##          Neg Pred Value : 0.6667
##              Prevalence : 0.8103
##          Detection Rate : 0.7931
##    Detection Prevalence : 0.9483
##       Balanced Accuracy : 0.5803
##
##        'Positive' Class : 0
##
```

4.3. Comparaison des prédictions

Les matrices de confusion montrent une une **accuracy** légèrement meilleure avec le modèle logistique qu'avec le modèle de Cox, même si les ordres de grandeur sont similaires.

Modèle	Accuracy	FP	VP	FN	VN
COX	0.7931	3	2	9	44
GLM	0.8276	1	2	9	46

De plus que la taille de l'échantillon de test est faible. On voit que la différence se fait seulement sur 2 faux positifs au lieu de 2 vrais négatifs.

5. Courbes ROC

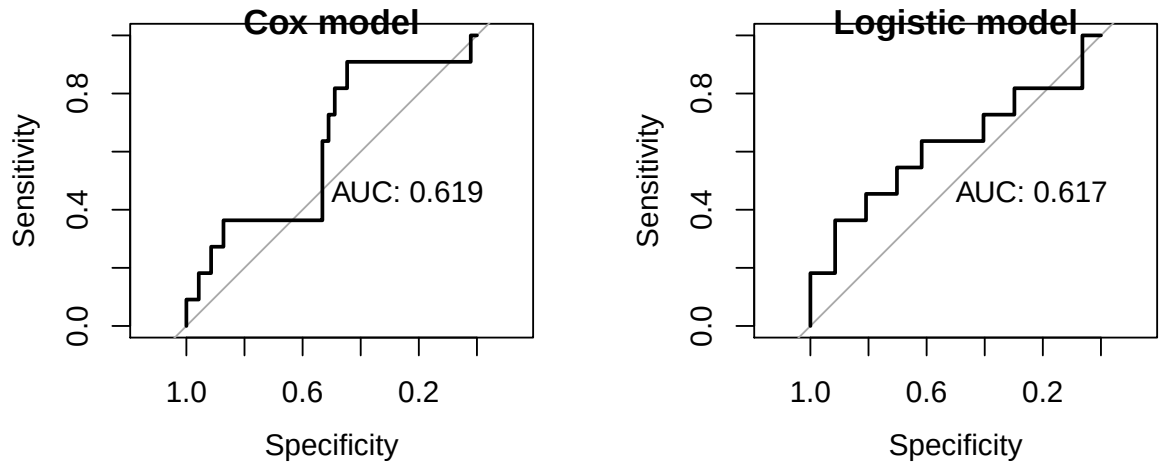
```
# Plan d'affichage
  par(mfrow=c(1,2))
# ROC : Cox model
  roc_cox = roc(test$label,cox_prediction,smoothed=T,ci=F,plot=TRUE,print.auc=TRUE)
  title("Cox model",outer = F)
# ROC : logistic model
  roc_glm = roc(test$label,glm_prediction,smoothed=T,ci=F,plot=TRUE,print.auc=TRUE)
  title("Logistic model",outer = F)
```

```
## Setting levels: control = 0, case = 1
```

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

```
## Setting levels: control = 0, case = 1
```

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



6. Conclusion

On a ici 2 modèles avec des performances à peu près équivalentes en terme d'AUC.
On peut noter des ruptures plus nettes de la courbe ROC pour le modèle de Cox.
Un plus grand echantillon et/ou du bootstrapping nous permettrait d'affiner cette comparaison.