kadaoui_alexandre_Rpart

Alexandre

23/12/2020

R Markdown

Dans le cadre de notre partiel, nous devons réaliser un total de 12 travaux retracant notre parcours et notre travail durant les 30 heures de cours.
Le travail à faire est le suivant :
- Une entête comportant un titre, un lien Github avec le ou les noms des auteurs.
- Une synthese de ce travail
- Un extrait commenté avec des parties de codes clé avec explication et commentaire.
- Une évalutation du travail avec nos 5 criteres.
- Une conclusion du travail

Definition des 5 critères de notations :

1) Effort de présentation :
2) Le knit est réalisable et bien présenté.
3) Explications simples et efficaces.
4) Le Code reproductible à d'autres DataFrame avec facilité.
5) Description des fonctions utilsés et du raisonnement.

Rpart / CART

Travail réalisé par "Maxime & Siva" le 11/11/2020.

https://github.com/mallaker/PSB_X/tree/main/Package%20Rpart
(https://github.com/mallaker/PSB_X/tree/main/Package%20Rpart)

Synthese :

RPART (Recursive Partitioning And Regression Trees) aussi connu sous le nom de CART (Classification And Regression Trees) est un package dont le but est de construire des modèles prédictifs.
Ces modèles prédictifs prennent la forme d'arbres de décision, s'appuyant sur plusieurs paramètres à chaque embranchement afin de déterminer la solution la plus adéquate possible à mesure que l'on avance vers les branches les plus extérieures de l'arbre.
Ces modèles peuvent être utilisés pour la classification et/ou la régression
Ce modèle d'apprentissage offre de par son design en arbres une bonne facilité et compréhension et d'interprétation de par son concept simple et sa représentation graphique concrète.
Cependant, cette méthode est limitée dans le cas de gestion de trop gros volumes de données.
Cette méthode est également très susceptible au sur-apprentissage (dans le cas où chaque branche finale de l'arbre correspondrait exactement à chaque unique cas du pool de données d'entrainement, avec ce que cela comporte de données aberrantes et de bruit). Afin d'éviter cela, il est nécessaire de pratiquer un "élagage" en supprimant un certain nombres de branches de l'arbre (si possible en retirant au maximum les valeurs aberrantes).

Extrait commenté du code :

Librairies nécessaires

```
library(rpart)
library(rpart.plot) #plot pour la represention de l'arbre de decision rpart
```

On utilise ici le dataset Ptitanic, contenant diverses information sur les passagers du Titanic ainsi que leur survie ou non suite au du nauffrage du navire.

```
data(ptitanic)
summary(ptitanic)#description des données
```

```
##  pclass        survived       sex           age              sibsp
##  1st:323   died    :809   female:466   Min.   : 0.1667   Min.   :0.0000
##  2nd:277   survived:500   male  :843   1st Qu.:21.0000   1st Qu.:0.0000
##  3rd:709                               Median :28.0000   Median :0.0000
##                                        Mean   :29.8811   Mean   :0.4989
##                                        3rd Qu.:39.0000   3rd Qu.:1.0000
##                                        Max.   :80.0000   Max.   :8.0000
##                                        NA's   :263
##      parch
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.385
##  3rd Qu.:0.000
##  Max.   :9.000
##
```

```
lapply(ptitanic,class) #donne la classe de chaque variable
```

```
## $pclass
## [1] "factor"
##
## $survived
## [1] "factor"
##
## $sex
## [1] "factor"
##
## $age
## [1] "labelled"
##
## $sibsp
## [1] "labelled"
##
## $parch
## [1] "labelled"
```

Ce morceau de code a pour but de vérifier que les variables “age” “sibsp” (nombre de frères, soeurs, mari/épouse à bord) et “parch” (nombre d’enfants ou parents à bord) sont bien des nombres Cependant la méthode ne m’est pas familière et aucune information additionnelle n’est fournie dans le document.

```
attr(ptitanic$age,"class") <- NULL
class(ptitanic$age)
```

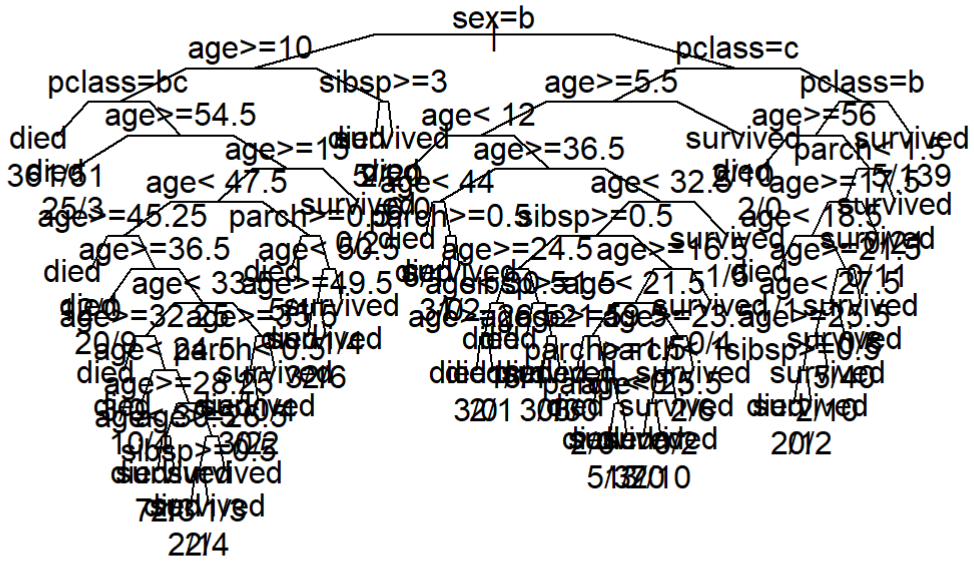
```
## [1] "numeric"
```

On sélectionne ici les 75% premières lignes du data set afin de servir de pool de données d'apprentissage. Les 25% constituant les dernières lignes du tableau seront utilisées plus tard afin de tester la précision du modèle.

```
nb_lignes <- floor((nrow(ptitanic)*0.75)) #on selctionne le nombre de ligne pour notre echant
illons d'apprentissage soit 75% du dataset initial
ptitanic.apprt <- ptitanic[1:nb_lignes, ]#echantillon d'apprentissage
ptitanic.test <- ptitanic[(nb_lignes+1):nrow(ptitanic), ]#echantillon de test
```

Construction de l'arbre avec la fonction **rpart()** grâce au pool de données d'apprentissage établi plus haut
On souhaite ici prédire la variable "survived", on la place donc à gauche du symbole "~"
On souhaite prédire cette variable en s'appuyant sur toutes les autres variables, on représente cela par un point. On placera ces variables de détermination à gauche du symbole "~"
rpart.control a pour but d'élaguer l'arbre afin d'éviter le sur-apprentissage
Ici chaque branche contenant 5 observations ou plus (cas trop spécifiques) seront coupés, cp=o désigne une absencce de contrainte de decoupage

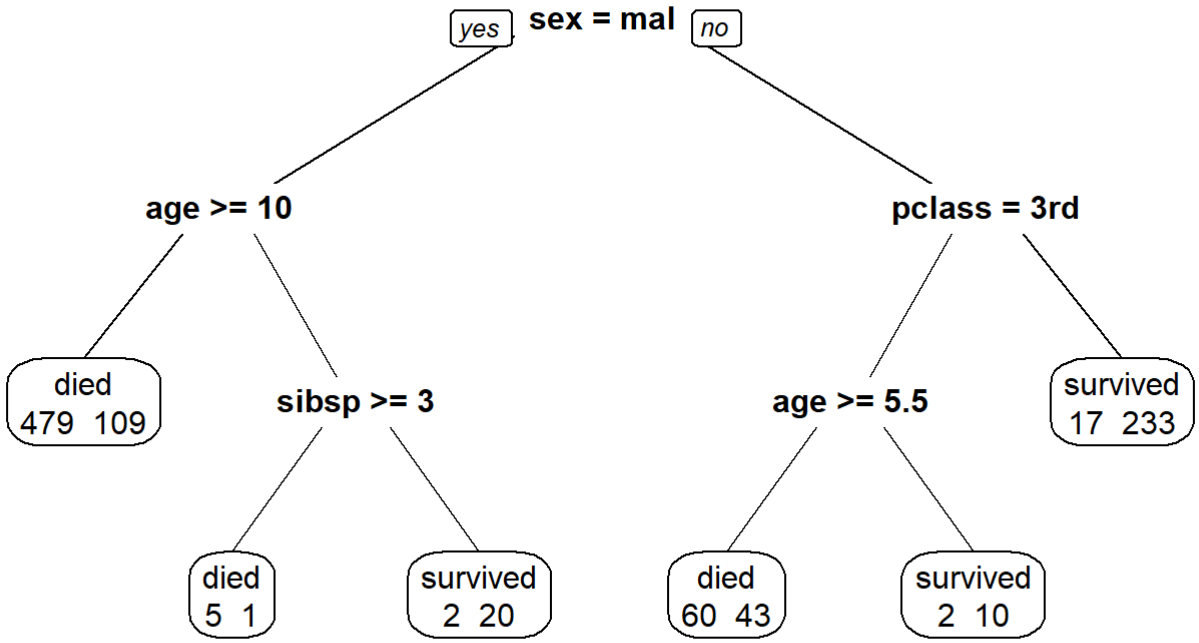
```
#construction de l'arbre
ptitanic.Arbre <- rpart(survived~.,data= ptitanic.apprt,control=rpart.control(minsplit=5,cp=0
))
#affichage de l'arbre
plot(ptitanic.Arbre, uniform=TRUE, branch=0.5, margin=0.1)
text(ptitanic.Arbre,all=FALSE, use.n=TRUE)
```



Elagage de l'arbre avec un CP idéal déterminé grâce à la fonction **which.min**

```
ptitanic.Arbre_Opt <- prune(ptitanic.Arbre,cp=ptitanic.Arbre$cptable[which.min(ptitanic.Arbre
$cptable[,4]),1])

#Affichage de l'arbre
prp(ptitanic.Arbre_Opt,extra = 1)
```



Prédictions du modèle pour le pool de données de test

```
#prediction du modele sur les données de test
ptitanic.test_predict <- predict(ptitanic.Arbre_Opt,newdata =ptitanic.test,type = "class")
#affichons juste la prediction faite sur les 10 premiers elements
print(ptitanic.test_predict[1:10])
```

```
##  982  983  984  985  986  987  988  989  990  991
## died died died died died died died died died died
## Levels: died survived
```

```
#Matrice de confusion
MC <- table(ptitanic.test$survived,ptitanic.test_predict)
print(MC)
```

```
##           ptitanic.test_predict
##            died survived
##   died      238        6
##   survived   77        7
```

Evaluation des performances du modèle

```
#Erreur de classement
erreur <- 1.0-(MC[1,1]+MC[2,2]/sum(MC))
print(erreur)
```

```
## [1] -237.0213
```

```
#Taux de prediction
prediction <- MC[2,2]/sum(MC[2,])
prediction
```

```
## [1] 0.08333333
```

Interprétation finale

```
print(ptitanic.Arbre_Opt)
```

```
## n= 981
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 981 416 died (0.57594292 0.42405708)
##    2) sex=male 616 130 died (0.78896104 0.21103896)
##      4) age>=10 588 109 died (0.81462585 0.18537415) *
##      5) age< 10 28   7 survived (0.25000000 0.75000000)
##       10) sibsp>=3 6   1 died (0.83333333 0.16666667) *
##       11) sibsp< 3 22   2 survived (0.09090909 0.90909091) *
##    3) sex=female 365  79 survived (0.21643836 0.78356164)
##      6) pclass=3rd 115  53 died (0.53913043 0.46086957)
##       12) age>=5.5 103  43 died (0.58252427 0.41747573) *
##       13) age< 5.5 12   2 survived (0.16666667 0.83333333) *
##      7) pclass=1st,2nd 250  17 survived (0.06800000 0.93200000) *
```

Evaluation du travail :

1) Effort de présentation : La présentation est bien soignée et parfaitement lisible et organisée
2) Le knit est réalisable et bien présenté : Le Knit s'effectue sans difficulté, tous les éléments à afficher du code s'impriment parfaitement bien.
3) Explications simples et efficaces : Le tutoriel prend le temps de détailler chaque étape dans l'ordre, ajouter des précisions sur tous les aspects du code et déroule le raisonnement d'une manière logique et fluide. Tous les aspects sont abordés de la manière la plus simple et compréhensible possible.
4) Le Code reproductible à d'autres DataFrame avec facilité : Grace aux nombreuses précisions et annotations expliquant toutes les fonctions, variables et leur fonctionnement, ce code est parfaitement reproductible et réutilisable dans de très nombreux cas pratiques différents de l'exemple vu ici.
5) Description des fonctions utilsés et du raisonnement : Les principales fonctions utilisées ici sont **rpart** et **rpart.control** dans le cadre de la formation/entrainement du modèle, la fonction **predict** pour l'application du modèle ainsi que la fonction **which.min** dans le cadre de l'élagage nécessaire afind d'éviter un sur-apprentissage et donc une sur-spécification du modèle.

Conclusion :

En conclusion, ce tutotriel extrêmement bien réalisé permet de comprendre de manière simple, rapide et visuelle la création de modèles prédictifs sous formes d'arbres de décision dans un but de classification ou de regression. La construction fluide et logique du tutoriel permet de bien suivre les étapes afin de s'approprier le

contenu/la méthode et d’être en mesure de l’appliquer à de futurs nouveaux cas.