RPART PACKAGE

Maxime & Siva

I. RPART, QU'EST CE QUE C'EST?

Le package RPART (Recursive Partitioning And Regression Trees) permet de construire des modèles de classification ou de régression d'une structure très générale en utilisant une procédure en deux étapes; les modèles résultants peuvent être représentés sous forme d'arbres de decision. Le package met en œuvre de nombreuses idées trouvées dans le livre et les programmes CART (Classification and Regression Trees) de Breiman, Friedman, Olshen et Stone.

En d'autre terme, la méthode de l' arbre de décision est une technique d'apprentissage automatique prédictif puissante et populaire qui est utilisée à la fois pour la classification et la régression . Ainsi, il est également connu sous le nom d' arbres de classification et de régression (CART).

Nous comprenons tout de suite que le modele sert à realiser un **arbre de decision** et comme son nom l'indique cela permet d'avoir toutes les options sur un graph afin de prendre une decision quelconque.

II. LE FONCTIONNEMENT DE L'ARBRE DE DECISION?

Les algorithmes de Decision, que l'on appelle également arbres de décision font partie de la catégorie des algorithmes supervisés, ils permettent de prédire une valeur (prédiction) ou une catégorie (classement).

C'est une méthode très populaire en Data Science et qui a donné naissance à d'autres algorithmes plus puissants tels que Random Forest ou XGBoost par exemple. Comme son nom l'indique, cet algorithme se base sur la construction d'un arbre ce qui rend la méthode assez simple à expliquer et plus facile à interpréter.

III. LES AVANTAGES ET INCONVENIENT D'UTILISER UN ARBRE DE DECISION

1. LES AVANTAGES

Facile à comprendre : le 1er avantage de cet algorithme c'est qu'il est intuitif, il est vraiment simple à comprendre. Et naturellement on a toujours tendance à préférer utiliser quelque chose que l'on comprend et que l'on maitrise

Facile à interpréter : son 2e avantage c'est qu'il est facile à interpréter. Les résultats peuvent être présentés à des équipes métier et les règles de décision produites par l'arbre sont faciles à comprendre.

Temps d'exécution raisonnable : un dernier avantage qui peut avoir son importance, c'est un algorithme assez simple qui n'est pas très coûteux en temps de calcul.

2. LES INCONVENIENTS

Faible performance : le principal inconvénient des arbres de décision est le manque de performance par rapport à d'autres algorithmes. Mais il est parfois nécessaire de faire un choix entre performance et interprétabilité.

Risque de sur-apprentissage : deuxième inconvénient ou en tout cas un piège dans lequel il ne faut pas tomber... L'overfitting c'est-à-dire le sur-apprentissage (l'algorithme apprend avec tellement de précision les données d'entrainement qu'il ne parvient pas à généraliser un résultat satisfaisant sur de nouvelles données). Pour éviter cela il est important de bien élaguer son arbre de décision.

IV. ELAGAGE

Un des inconvénients de l'utilisation des arbres de decision c'est le risque de sur-apprentissage. On peut laisser l'algorithme se déployer au maximum pour coller parfaitement au dataset d'entrainement (échantillon

des données sur lequel on entraine l'algorithme). Dans ce cas là la performance de l'algorithme sur les données qui ont permis de le créer sera excellente. En revanche il se généralisera très mal sur de nouvelles données. Ce n'est pas ce que nous cherchons.

Pour éviter cela il est nécessaire d'avoir un arbre par trop grand. C'est là que l'élagage intervient. Cette méthode consiste simplement à supprimer des branches et des feuilles de l'arbre.

Nous allons passer aux choses serieuses et donc à la pratique pour comprendre comment on utilise l'arbre de decision.

Chargement des librairies

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

```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

Chargement des données

Nous allons utiliser un dataset denommé Ptitanic fourni dans R avec la librairie Rpart, un jeu de données concernant le nauffrage d'un titanic et a bord des individus classés par categories (variables), le fichier contient 1309 individus et 6 variables qui sont : **pclass** : donne la classe tarifaire sur le bateau ; **survided** : indique si le passager a survécu ; **sex** : donne le genre du passager ; **age** : donne l'âge du passager exprimé en années ; **sibsp** : donne le nombre de frères, sœurs, mari ou épouse à bord ; **parch** : donne le nombre d'enfants ou de parents à bord.

```
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"
```

Nous avons donc trois variables nominales qui sont bien des “factor” et pour verifier que les autres variables sont numeriques, on procede de la maniere suivante:

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

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

V. CREATION D’UN DATASET D’APPRENTISSAGE

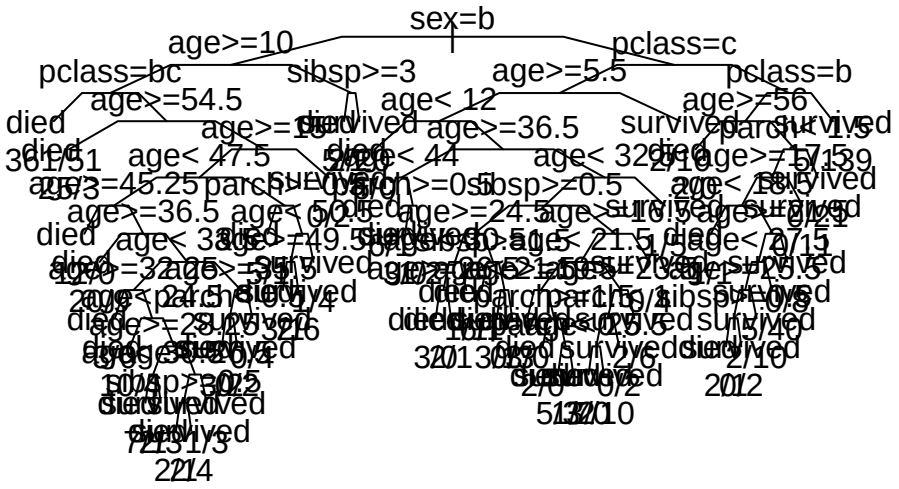
Comme dans tout modele d’apprentissage d’algorithme, on crée un dataset d’apprentissage sur lequel on entrainera notre algorithme ensuite un dataset de validation pour tester l’algorithme afin d’evaluer sa performance.

```
nb_lignes <- floor((nrow(ptitanic)*0.75)) #on selctionne le nombre de ligne pour notre echantillons d'a

ptitanic.apprt <- ptitanic[1:nb_lignes, ]#echantillon d'apprentissage
ptitanic.test <- ptitanic[(nb_lignes+1):nrow(ptitanic), ]#echantillon de test
```

VI. CONSTRUCTION DE L’ARBRE DE DECISION

Nous allons proceder à la construction de l’arbre de decision en utilisant l’echantillon d’apprentissage.

```
#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)
```

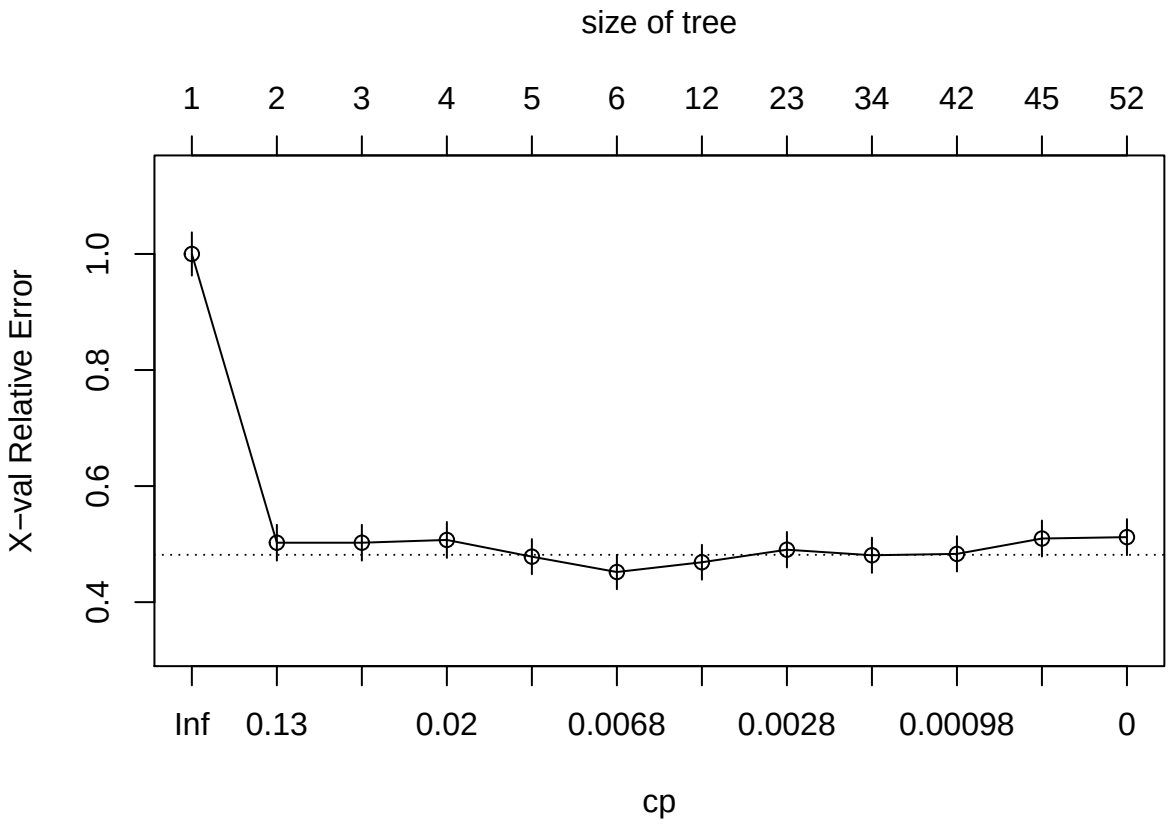


La formule utilisée survived~. indique qu'on souhaite prédire la variable survived en fonction de toutes les autres. Le principe général est que la (ou les) variable(s) à prédire sont à gauche du symbole ~ alors que les variables prédictives sont à droite du symbole. Ici, le point . permet d'indiquer qu'on souhaite utiliser toutes les variables des données comme prédicteurs sauf les variables à prédire. La commande rpart.control permet preciser les elements à modifier dans la construction de l'arbre, le minsplit permet de decouper les feuilles qui contiennent au moins 5 observations, cp=0 pour dire sans contrainte de decoupage.

Niveaux de simplification

On voit que l'arbre est surchargé d'information au niveau des feuilles (information illisible) et l'interpretation de cet arbre est compliqué du coup on va proceder à l'elagage pour supprimer certaines feuilles et organiser les informations.

```
plotcp(ptitanic.Arbre)
```



Le graphique ci-dessus nous permet de choisir le cp optimal (la complexité qui minimise l'erreur estimée) pour notre arbre de decision du fait que lorque le nombre de feuilles augmente, l'arbre se degrade en raison du sur-apprentissage.

On va afficher le CP otpimal

```
print(ptitanic.Arbre$cptable[which.min(ptitanic.Arbre$cptable[,4]),1])
```

```
## [1] 0.004807692
```

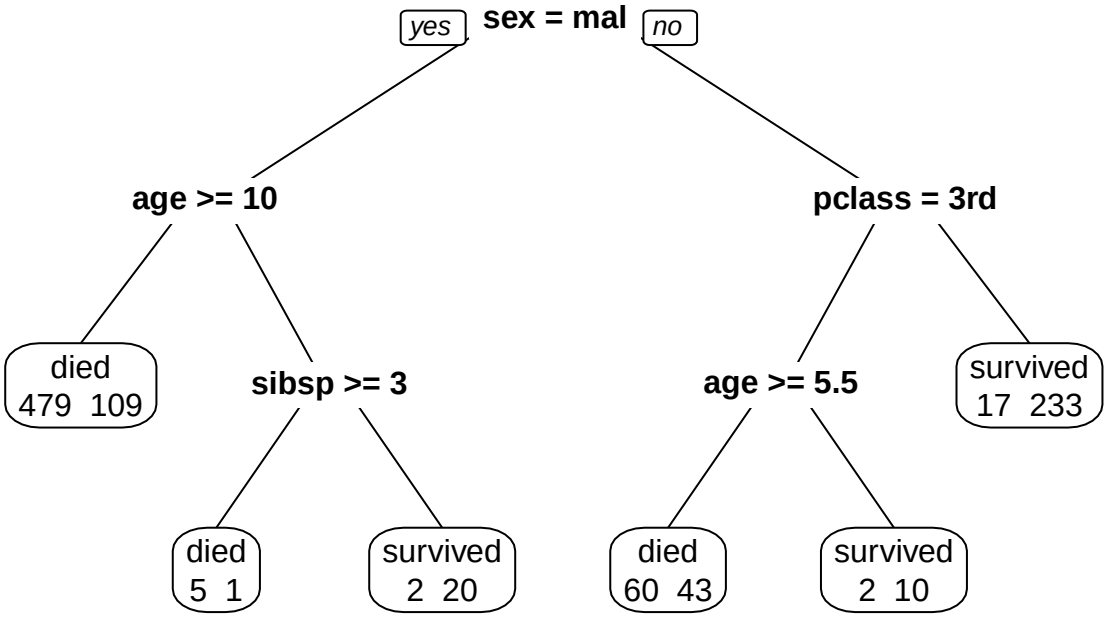
les valeurs contenues dans la matrice avec la fonction cptable sont celles qui produisent des changements dans le nombre de feuilles de l'arbre.

Elagage de l'arbre avec notre Cp optimal obtenu

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

Representation gragphique de l'arbre optimal

```
prp(ptitanic.Arbre_Opt,extra = 1)
```



Interpretation : Ici, pour chaque feuille, R indique la classe prédite, le nombre d'individus de la classe prédite à gauche et le nombre d'individus de l'autre (ou des autres) classes à droite. Par exemple, si on prend la première feuille à gauche qui correspond au sexe masculin ensuite par categorie d'age (superieur ou égal à l'age inndiqué), la feuille contient le chiffre à droite indiquant le nombre des passagers décédés et le chiffre à droite le nombre de passagers survivants, et cette sous-population est classée dans « Died ».

VII. TEST ET VALIDATION

On va test et valider les resultats avec l'echantillon de test

1. Prevision

```
#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
```

2. Evaluation des performance

```
#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
```

Le taux de prediction etant acceptable, on peut afficher le resultat et proceder à une interpretation 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) *
```

Chaque ligne correspond à un noeud ou à une feuille de l'arbre. On commence par la racine qui contient les 981 individus de l'échantillon d'apprentissage. le taux moyen general de survie pour tout l'echantillon est de 38.53% contre 60.55% de decés.

A l'étape suivante l'arbre découpe la population en fonction de la variable « Sex » et créé les noeuds 2) et 3). Le noeud numéro 3 contient les femmes : 362 individus. Leur taux de survie est de l'ordre de 73% et est donc supérieur au taux moyen de 38.53%. Ce noeud a donc été labellisé comme « Survived ». R indique ensuite le nombre d'individus mal classés dans ce noeud : 96 (femmes qui sont décédées). Nous avons ensuite le taux de mauvais classement : 27% et le taux de bon classement : 73%.

On continue ainsi de suite le découpage de l'arbre jusqu'aux feuilles qui sont identifiées par une étoile en fin de ligne.

Bien sûr il est interdit d'utiliser ce type de représentations dans les présentations de résultats, c'est trop technique. Il vaut mieux digérer toutes ces informations et les restituer de manière plus visuelle.