RPART PACKAGE de Maxime & Siva

Tarik HAKAM

22/12/2020

1. Critères d'évaluation

1. Comportement du Rmd à l'exécution
2. Qualité de la rédaction du dossier
3. Accessibilité, didactisme et pertinence du dossier
4. Qualité et lisibilité du Rmarkdown
5. Qualité des applications permettant d'illustrer le package

2. Lien vers le document commenté

En cliquant **ici**, vous trouverez le lien menant au GitHub de Maxime hébergeant le fruit de sa collaboration avec Siva.

3. Auteurs du document commenté

Le document évalué dans le cadre de ce rendu a été produit par Maxime & Siva, étudiants en Master of Science Data Management à Paris School of Business.

4. Synthèse du document

Ce document présente le package RPART (Recursive Partitioning And Regression Trees).

Celui-ci 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 décision.

Cette méthode est une technique d'apprentissage automatique prédictive puissante et populaire connue également sous le nom **d'arbres de classification et de régression** (CART).

Cette dernière a donné naissance à d'autres algorithmes plus puissants tels que ***Random Forest*** ou ***XGBoost***.

Comme son nom l'indique, cet algorithme se base sur la construction d'un arbre qui rend la méthode assez simple à expliquer et plus facile à interpréter.

Les arbres de décision font partie de la catégorie des algorithmes supervisés, ils permettent de prédire une valeur (prédiction) ou une catégorie (classement).

1. Ses avantages

- Cet algorithme est **facile à comprendre** et **intuitif**
- Il est **facile à interpréter**
- Le **temps d'exécution est raisonnable** (algorithme peu coûteux en terme de temps de calcul).

2. Ses inconvénients

- Les arbres de décison ont une **faible performance** par rapport à d'autres algorithmes.
- L'algorithme tend vers un **risque de sur-apprentissage** (ou overfitting). L'algorithme apprennant, avec une très grande précision, les données d'entrainement, cela induit, de ce fait, un risque ne permettant pas de généraliser un résultat satisfaisant sur de nouvelles données. Pour éviter ce phénomène, il est nécessaire d'élaguer l'arbre de décision. Cette méthode consiste tout simplement à supprimer des branches et des feuilles de l'arbre.

5. Extrait commenté des parties de code

Ici, il a été utilisé le dataset ***Ptitanic*** disponible avec la librairie Rpart. Ce jeu de données concerne le naufrage d'un Titanic avec à son bord des individus classés par catégorie (variables). Le fichier recense 1309 individus et les 6 variables suivantes :

- **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.

Dans un 1^{er} temps, les auteurs créent un dataset d'apprentissage sur lequel l'algorithme s'entrainera, ainsi qu'un dataset de validation pour tester l'algorithme afin d'évaluer sa performance.

```
nb_lignes <- floor((nrow(ptitanic)*0.75))
ptitanic.apprt <- ptitanic[1:nb_lignes, ]
ptitanic.test <- ptitanic[(nb_lignes+1):nrow(ptitanic), ]
```

Ici, ils sélectionnent un nombre de lignes à partir du dataset initial, soit 75% de ce dernier afin de définir l'échantillon d'apprentissage, ***ptitanic.apprt***. Puis, ils définissent l'échantillon de test, ***ptitanic.test***, soit 25% du dataset initial.

L'étape suivante consiste à construire un arbre de décision en utilisant l'échantillon 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 que les auteurs souhaitent prédire la variable *survived* en fonction de toutes les autres.
- La commande `rpart.control` permet de préciser les éléments à modifier dans la construction de l'arbre.
- Le `minsplit` permet de découper les feuilles qui contiennent au moins 5 observations.
- Le `cp=0` permet de signifier qu'aucune contrainte de découpage n'est exercée.

Suite à l'affichage de l'arbre de décision, les auteurs souhaitent ensuite procéder à l'élagage pour supprimer certaines feuilles et organiser les informations. Au vu de la surcharge d'information de ce dernier, l'interprétation est difficile et la lecture illisible.

Pour ce faire et à travers le morceau de code ci-dessous, les auteurs recherchent le cp optimal (la complexité qui minimise l'erreur estimée) pour cet arbre de décision dont la lecture a été degradée faute à un sur-apprentissage.

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

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

Une fois le cp optimal identifié, il peuvent procéder à l'élagage de l'arbre avec le cp optimal obtenu.

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

A l'issue de cette manipulation, il est obtenu un arbre "optimisé".

Ainsi, les auteurs ont réalisé le test et la validation des résultats avec l'echantillon de test.

6. Evaluation du travail suivant les 5 critères précités

1. Comportement du Rmd à l'exécution

L'exécution du RMarkdown se réalise parfaitement.

2. Qualité de la rédaction du dossier

La rédaction de ce document est de bonne qualité. Le language est adapté et il existe un fil conducteur dans la construction de la pensée des auteurs.

3. Accessibilité, didactisme et pertinence du dossier

La lecture de ce dossier est dans son ensemble aisée et accessible au plus grand nombre, même aux néophytes. Les descriptions, dans la première partie (construction d'un arbre de décison), sont relativement bien explicitées et bien illustrées. Cependant, la partie "Test et Validation" est moins documentée, plus technique et plus difficile à comprendre sans recherche personnelle.

Les auteurs arrivent cependant à faire adhérer le lecteur à leur production. Le document a une réelle vocation à transmettre une connaissance et le but, pour ma part, est atteint dans son ensemble (2ème partie à compléter).

Le sujet est pertinent et a vocation à servir de mémo pour de futures réalisations.

4. Qualité et lisibilité du RMarkdown

Le RMarkdown est globalement bien écrit, lisible et aéré.

J'aurai éventuellement quelques suggestions :

1. J'aurais personnellement inséré quelques sauts de lignes afin d'aérer le document PDF.
2. Ligne 80 du code Rmd : le commentaire dépasse de la cellule et est tronqué dans le document PDF final. Il aurait été intéressant de le réorganiser.
3. Idem pour les lignes de code 93 et 122.

Autrement, il s'agit d'un code bien réalisé.

5. Qualité des applications permettant d'illustrer le package

Les applications sont, dans la première partie (la plus conséquente), de bonne qualité et bien expliquées. Quelques doutes cependant autour de la partie **"Test et Validation"** concernant la maitrise des notions. Ce sentiment est exacerbé par le manque de guidance dans l'explication.

7. Conclusion

Selon moi, il s'agit d'un bon travail.

Les deux auteurs ont dans l'ensemble bien documenté leur dossier. C'est un travail bien construit, recherché, a visée didactique, avec un effort réel pour le rendre accessible au plus grand nombre.

Très bien.

Vous retrouvez ce document sur mon **GitHub**.