Résumé sur le package Caret

Jingwen SU

12/20/2020

Introduction

À la fin du cours, tous les étudiants ont téléchargé les documents pertinents sur Github et partagé les résultats avec nous. Afin d'utiliser ces packages d'installation de R de manière plus complète, je vais lire, analyser et évaluer les articles de mes collègues pour améliorer mes lacunes.

Ceci est une introduction à l'article lu dans cet article. Si vous souhaitez en savoir plus, vous pouvez rechercher des références.

- **Title:** Qu'est ce que le CARET ?
- **Auteurs:** Théo Marié
- **Lien sur Github:** Théo Marié

Synthèse du travail en question

Dans cet article, l'auteur a créé une atmosphère d'apprentissage détendue, étape par étape. Tout d'abord, il nous a présenté ce qu'est le caret et a expliqué les principes de l'analyse prédictive. Ensuite, il a commencé à nous expliquer comment installer le package d'installation caret et appeler les fonctions qu'il contient pour créer un tableau. Après cela, il a vérifié les résultats du tableau via le modèle de forêt aléatoire.

Contenu principal et explication

1.Installez et créez un tableau de données

Pour installer le caret, utilisez la fonction familière **library(caret)**

Afin de garantir le caractère aléatoire des données, nous devons utiliser la fonction **seed** avant de créer le tableau.

Pour la création du tableau de données, nous utiliserons la fonction **createDataPartition**.

```
library(caret) ## ON INVOQUE CARET
set.seed(333) ## ON PLANTE UNE GRAINE

trainIndex <- createDataPartition(iris$Species, p = 0.8,
                                  list = FALSE,
                                  times = 1)

irisEntrainement <- iris[ trainIndex,] ##Création du dataset d'entrainement
irisTest <- iris[-trainIndex,]##Création du dataset de Test
```

2.Modélisation

L'auteur donne une idée de modélisation claire. Avant de construire le modèle, nous utiliserons la fonction **train**, où le modèle est une forêt aléatoire **rf**.

```
Theo <- train(Species ~ .,
              data = irisEntrainement,
              method = "rf",) ##il faut ici installer le package randomforest
```

3.Sortez le résultat

```
# reussite <- tableau_de_prediction[,1] == tableau_de_prediction[,2]
# nombre_de_reussites <- sum(reussite)
# nombre_de_ligne <- nrow(tableau_de_prediction)
# exatitude_de_la_prediction <- nombre_de_reussites/nombre_de_ligne * 100
```

Ici, nous devons prêter attention à la pensée logique. Nous devons utiliser et construire des fonctions de manière flexible pour obtenir le résultat que nous voulons exprimer. Bien que nous ne puissions pas nous souvenir de toutes les fonctions, la manière d'atteindre l'objectif n'est pas la seule. nous devons être doués pour utiliser des affectations simples et d'autres méthodes.

Evaluation et résumer

Tout d'abord, l'avantage de cet article est que la technique d'écriture de l'auteur est très détendue, il n'a pas donné d'introduction rigide et donné des exemples. Au lieu de cela, le lecteur est toujours guidé pour apprendre en posant des questions. Et tout l'exemple est très complet. Les parties importantes de l'article sont marquées en rouge. Cependant, il a aussi des problèmes de formatage. Toutes les parties rouges de l'article ont "** **". Je pense qu'il pourrait vouloir l'agrandir, mais il ne l'a pas utilisé correctement. De plus, bien qu'il ait donné des exemples de fonctions, il n'a pas montré de résultats pertinents. Cela empêchera les lecteurs de voir visuellement l'effet de la fonction et affectera même leur compréhension.