Tutoriel caret

Tarik HAKAM

10/12/2020

Introduction

L'arbre de décision, ou Decision tree, est une méthode prédictive de Machine Learning éprouvée utilisée pour la classification et la regression.

Cette méthode est aussi connue sous le nom : Classification and Regression Trees (CART).

A noter que l'implémentation en R de l'algorithme CART est nommé RPART (Recursive Partitioning And Regression Trees), disponible sous ce même nom dans le package caret.

```
library("caret")
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
data(iris)
```

L'algorithme fonctionne en partitionnant de manière répétée les données en plusieurs sous-espaces, de sorte que les résultats dans chaque sous-espace final soient aussi homogènes que possible.

Cette approche est techniquement appelée partitionnement récursif.

Nous allons répartir les données de manière aléatoire en un training set (80% pour la construction d'un modèle prédictif) et un test set (20% pour l'évaluation du modèle).

```
#Modèle : Decision Tree
trainIndex <- createDataPartition(iris$Species, p = 0.8,
                                  list = FALSE,
                                  times = 1)
train <- iris[ trainIndex,]
test  <- iris[-trainIndex,]
```

Il faut mettre en place le set seed pour s'assurer de la reproductibilité.

```
set.seed(999)
```

Le résultat produit consiste en un ensemble de règles utilisées pour prédire la variable de résultat, qui peut être soit :

- une variable continue, pour les arbres de régression

- une variable catégorielle, pour les arbres de classification (ici notre cas)

On entraine le modèle :

La matrice de confusion est l'une des nombreuses façons d'analyser la précision d'un modèle de classification.

Comme le montre le tableau ci-dessous, une matrice de confusion est essentiellement un tableau bidimensionnel à deux axes.

Sur un axe, elle comporte des catégories réelles ou cibles et sur l'autre axe, elle contient des catégories prédites. Les cellules diagonales indiquent les vrais positifs, c'est-à-dire le nombre de cas tests qui ont été correctement prédits par le modèle.

```
#build confusion matrix
predictions <- predict(decision_tree, test)
confusionMatrix(predictions, test$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         2
##   virginica       0          0         8
##
## Overall Statistics
##
##                Accuracy : 0.9333
##                  95% CI : (0.7793, 0.9918)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 8.747e-12
##
##                   Kappa : 0.9
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.8000
## Specificity                 1.0000            0.9000           1.0000
## Pos Pred Value              1.0000            0.8333           1.0000
## Neg Pred Value              1.0000            1.0000           0.9091
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.2667
## Detection Prevalence        0.3333            0.4000           0.2667
## Balanced Accuracy           1.0000            0.9500           0.9000
```

```
confusion_matrix <- as.data.frame(table(predictions, test$Species))
names(confusion_matrix) = c("Actual","Predicted","Freq")

#compute frequency of actual categories
actual = as.data.frame(table(test$Species))
names(actual) = c("Actual","ActualFreq")
```

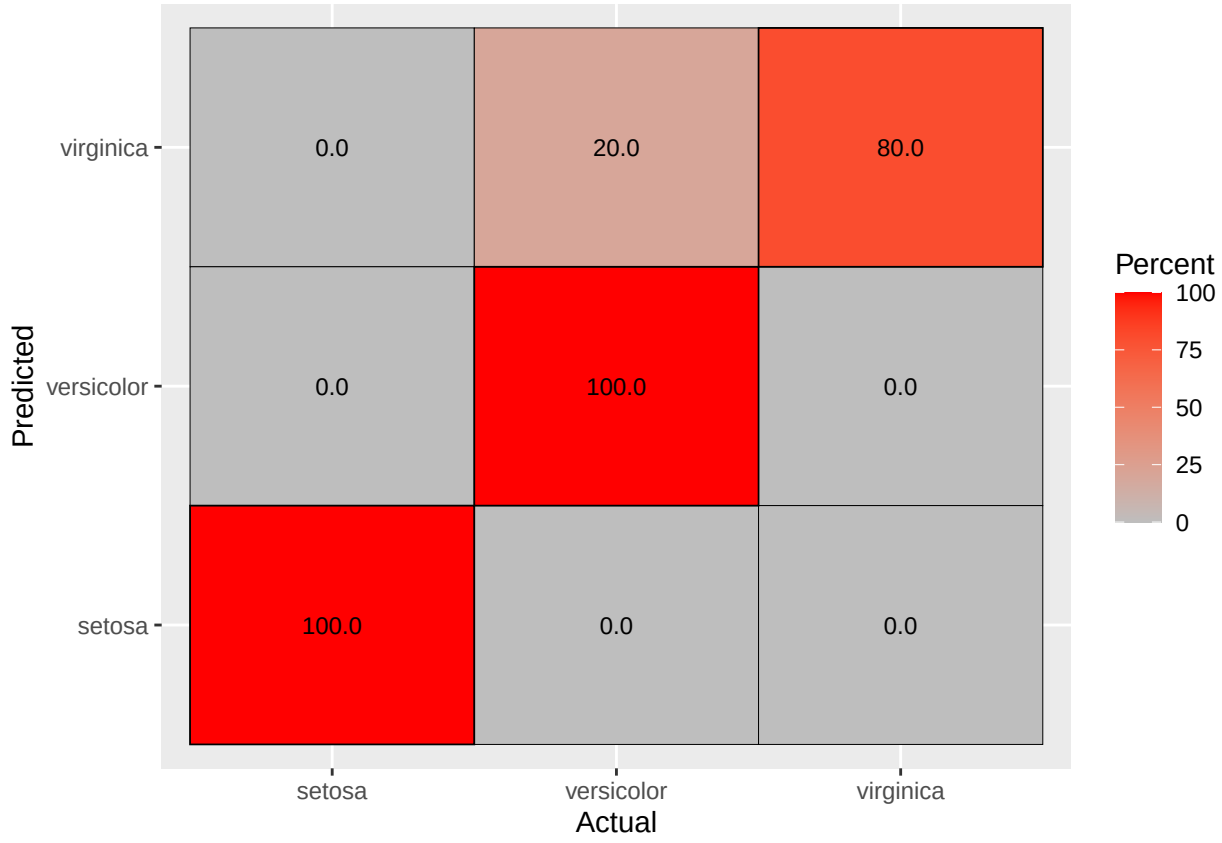
```
#calculate percentage of test cases based on actual frequency
confusion = merge(confusion_matrix, actual, by=c("Actual"))
confusion$Percent = confusion$Freq/confusion$ActualFreq*100

#render plot
# we use three different layers
# first we draw tiles and fill color based on percentage of test cases
tile <- ggplot() +
  geom_tile(aes(x=Actual, y=Predicted,fill=Percent),data=confusion,
            color="black",size=0.1) + labs(x="Actual",y="Predicted")

tile = tile +
  geom_text(aes(x=Actual,y=Predicted, label=sprintf("%.1f", Percent)),
            data=confusion, size=3, colour="black") +
  scale_fill_gradient(low="grey",high="red")

# lastly we draw diagonal tiles.
# We use alpha = 0 so as not to hide previous layers but use size=0.3 to highlight border
tile = tile +
  geom_tile(aes(x=Actual,y=Predicted),
            data=subset(confusion,as.character(Actual)==as.character(Predicted)),
            color="black",size=0.3, fill="black", alpha=0)

#render
tile
```



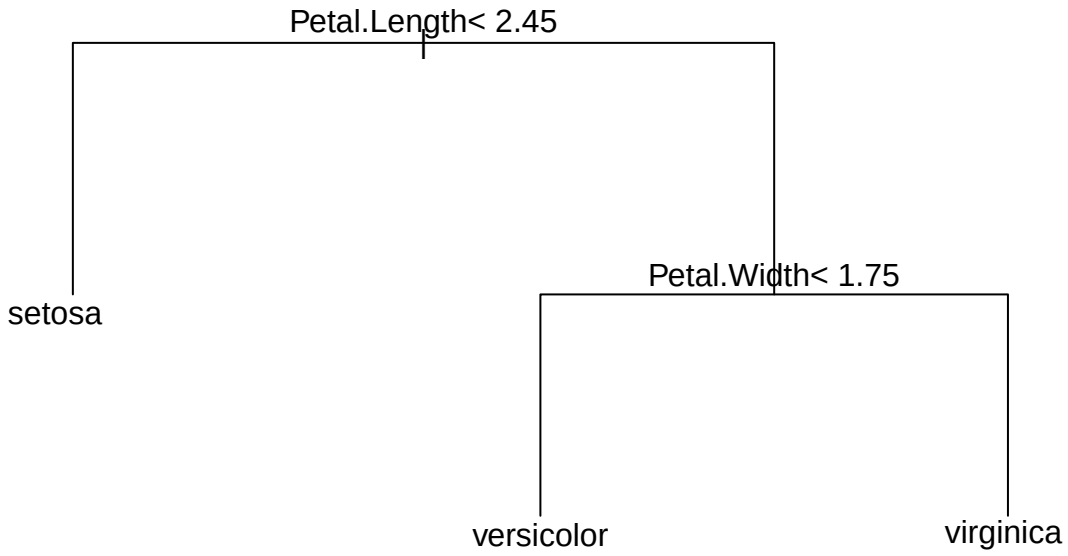
Les espèces setosa et les versicolor ont été prédites à 100%.

Seule l'espèce virginica n'a pas été totalement bien prédite : 20% des virginica ont été prédit comme des versicolor.

Les règles de décision générées par le modèle prédictif CART sont généralement visualisées sous la forme d'un arbre binaire, "bi-feuille".

L'exemple suivant représente un modèle d'arbre prédisant l'espèce de fleur d'iris en fonction de la longueur (en cm) et de la largeur du sépale et du pétale.

```
library(rpart)
model <- rpart(Species ~., data = iris)
par(xpd = NA) # otherwise on some devices the text is clipped
plot(model)
text(model, digits = 3)
```



La validation croisée, ou Cross Validation, est une procédure de rééchantillonnage utilisée pour évaluer des modèles de Machine Learning sur un échantillon de données.

La procédure comporte un seul paramètre k qui fait référence au nombre de classes dans lesquels un échantillon de données doit être divisé (ici k = 3 car 3 espèces d'iris).

Il existe de nombreuses mesures différentes que vous pouvez utiliser pour évaluer vos algorithmes en R.

Ainsi, pour évaluer vos modèles, les mesures par défaut utilisées sont l'accuracy pour les problèmes de classification et la RMSE pour la régression.

Mais caret prend en charge une série d'autres mesures d'évaluation populaires.

Vu notre cas, nous allons voir l'accuracy et Kappa.

Ce sont les mesures par défaut utilisées pour évaluer les algorithmes sur les ensembles de données de classification binaire et multiclasses en caret.

L'Accuracy est le pourcentage d'individus correctement classées parmi toutes les individus. Elle est plus utile pour une classification binaire que pour les problèmes de classification multiclasses, car il peut être difficile de savoir exactement comment l'accuracy se répartit entre ces classes (ie. comme nous avons vu plus tôt avec la matrice de confusion).

La méthode Kappa ou Kappa de Cohen est semblable à l'accuracy de la classification, sauf qu'elle est normalisée au hasard sur l'ensemble des données. Il s'agit d'une mesure plus utile à utiliser pour les problèmes qui présentent un déséquilibre dans les classes (par exemple, la répartition les virginica 80%-20%).

Ainsi, on peut voir les tableaux de l'accuracy et de Kappa pour l'algorithme du Decision Tree. Cela comprend les valeurs moyennes (à gauche) et les écarts types (marqués SD pour Standard Deviations) pour chaque mesure, repris dans la population des cross validation folds et trials.

Vous pouvez voir que l'accuracy du modèle est d'environ 93,33%. Le modèle Kappa, en revanche, affiche une précision d'environ 90%.

```
#Cross validation
fitControl <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 5)

#Train the model
decision_tree <- train(Species ~., data = train, method = "rpart",
                trControl = fitControl)
#Summarize the results
decision_tree
```

```
## CART
##
## 120 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
## Resampling results across tuning parameters:
##
##   cp    Accuracy   Kappa
##   0.00  0.9466667  0.92
##   0.45  0.7800000  0.67
##   0.50  0.3333333  0.00
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.
```