# "Qu'est ce que le CARET ?".

Théo Marié

Décembre 2020

---

Sur R, il est possible de faire de l'analyse prédictive.

Comment ? grace au package **Caret** (Classification And Regression Training) ! Pour se faire, il faut apprendre à bien uiliser le package, qui va être d'une grande utilité dans le management et le traitement de données.

Le pincipe de l'analyse prédictive se résume à :

**1** On analyse 80% des données, à l'aide d'une graine, **(seed)**. En pratique, on fait tourner l'algorythme sur 80% des données dont on dispose, afin d'en déduire une prédiction.

**2** Ensuite, une fois cette prédiction obtenue, on la vérifie en appliquant la prédiction obtenue précédemment aux 20% restant.

---

Commençons par le commencement : L'installation du package :

install.packages("caret")

Ensuite, bien évidemment, il faut l'appeler, l'invoquer, avec library.

**CreateDataPartition** va nous permettre de créer un tableau de datas, et si on se penche sur les paramètres qu'on lui attribue, on voit que p = 0.8, autrement dit que 80% des données du tableau vont nous permettre de s'enraîner, ce qui représente 120 Valeurs. Les 20% restant representeront donc 30 Observations.

IMPORTANT !!!

Vous venez de voir **set.seed**. C'est une étape cruciale.

**SEED** veut dire graine en anglais.Pour faire simple, c'est la graine que je vais donner à l'algorythme pour commencer ses calculs. La garaine, est la référence de départ que je vais donner à l'algorythme, pour générer l'aléat. L'aléatoire est nécessaire, sinon on peut se retrouver avec des valeurs qui ne s'afficheront pas, si dans les 20% restants des valeurs, certaines ne sont pas encore affichées.

```
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
```

Prochaine étape, **on va lancer l'entraînement !**

On va entraîner le modèle sur les datas d'iris, les fameux 80%. Il faut lui trouver un nom, et vu qu'il fait la liaision entre les species et les colonne de gauches, on va l'appeler **Theo**, car le bon Theo, il fait des liens entre les gens, il les rapproche.

Première question : Quelle est la cible ? ici c'est les Species, on veut pouvoir déterminer le modèle sur les espèces.

Pour utiliser toutes le colonnes, on s'aide du ** ** (tilde).

Deuxième question : Quel modèle utilisera-t-on ? PLusieurs modèles existent, comme la méthhode des arbres de descisions, des forêts, mais ici on utilisera la méthode forêt d'arbres : Au lieu de faire un seul arbre, on en prend plusieurs (forêt), puis on va faire la moyenne. Cela réduit la possibilité d'erreur, c'est donc plus solide, plus sûr. On retient donc random forest, rf (pensez à le télécharger!!)

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

Puis on applique le modèle Theo aux jeux de test **irisTest**, celui qui représente 20% ! On va donc avoir une prédiction, appelée \textcolor{red}{**prediction-du__test__iris**}.

Comment comparer les résultats? Comment comparer la prédiction que l'on a faite avec les 20% restant d' **irisTest**? On va simplement créer un tableau avec dataframe, pour juxtaposer une par une les valeurs Species. Il s'appelle alors \textcolor{red}{**tableau__de__prediction**}.

```
prediction_du_test_iris <- predict(Theo, irisTest)## il faut ici installer le package e1071
tableau_de_prediction <- data.frame(irisTest$Species, prediction_du_test_iris )
```

NOUS Y SOMMES PRESQUE !

Il nous manque plus qu'à faire de simples calculs pour le pourcentage de fiabilité de notre algorythme. On a donc quelques étapes à mettre en place :

Tout d'abord, on va définir lorsqu'une prédiction est **vérifiée**, c'est à dire quand l'algorythme affiche le même contenu entre la prédiction et le test : on l'appellera **réussite**. Ensuite, on calcule la **somme** de ces réussites. Après, il suffit de connaître **le nombre de lignes** qui ont été testées, puis **d'évaluer la proportion** de réussite de l'algorythme !

```
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 l'exactitude de la prédiction de cet algorythme est de **93,33%**.

**GOOD JOB**