Rapport TP2 - Partie C

Hubert Hirtz, Camille Schnell

10 décembre 2018

Objectif

Il s'agit ici, à partir du dataset *spam* (librairie **ElemStatLearn**), de comparer les performances de différentes machines, basées sur les modèles étudiés en cours.

Mise en œuvre

Pour effectuer la comparaison, nous utilisons une fonction de map. Pour chaque modèle, nous ....**expliquer train, test, et ce qu'on fait avec predict, mean**....

```
#chargement des librairies
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```
library(MASS)
library(tidyr)
library(rsample)
```

```
## Warning: replacing previous import by 'tibble::as_tibble' when loading
## 'rsample'

##
## Attaching package: 'rsample'

## The following object is masked from 'package:tidyr':
##
##     fill
```

```
library(rpart)
library(tree)
library(e1071)
library(ipred)
library(ElemStatLearn)
library(purrr)
library(tibble)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

```
#réinitialisation des données dans rstudio
rm(list=ls());
```

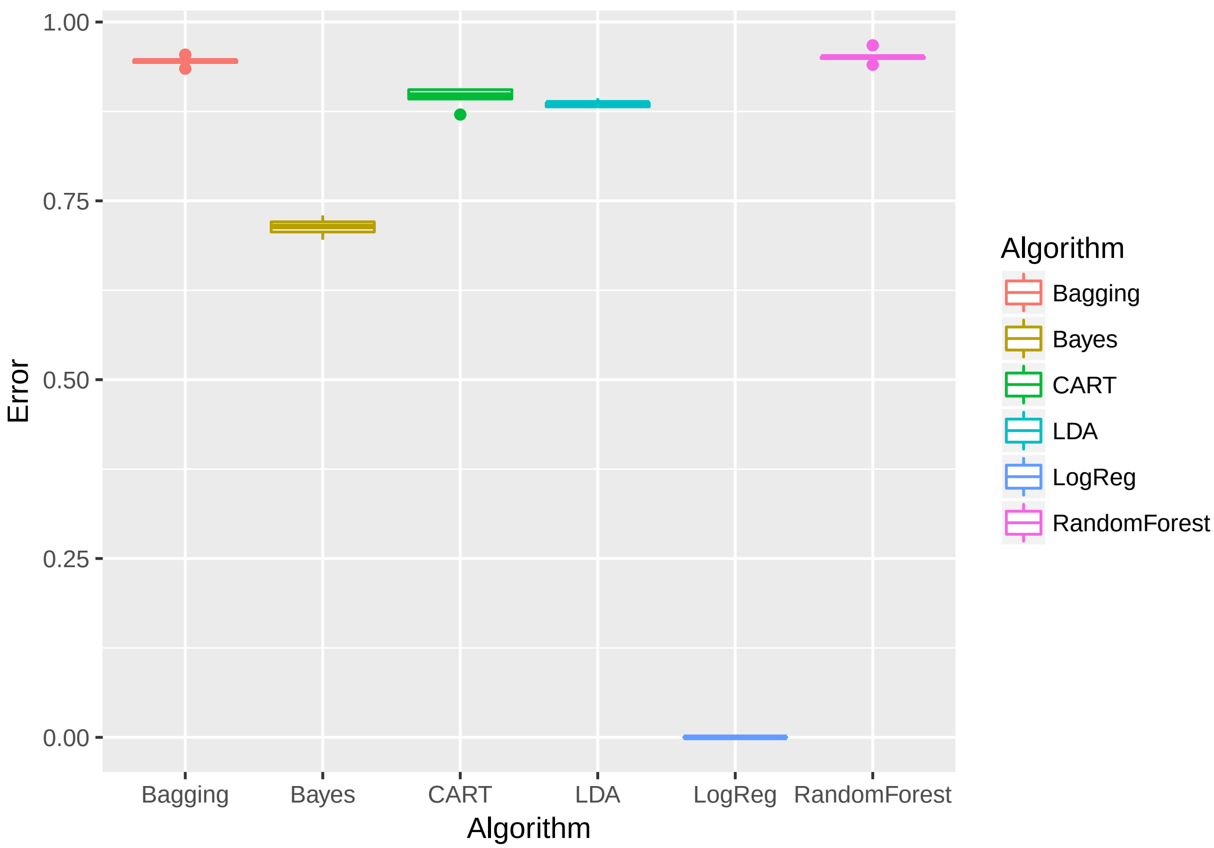
```r
graphics.off();

#récupération des données de spam
data(spam)

#analyse des performances
folds <- vfold_cv(spam, v=5)
performance <- map_dfr(folds$splits,
       function(x){
         x.train <- as_tibble(x, data = "analysis")
         x.test <- as_tibble(x, data = "assessment")

         #CART
         y_pred <- predict(tree(spam~., data = x.train), newdata = x.test,
                           type = "class")
         Tree <- mean(y_pred == x.test$spam)

         #Bagging
         y_pred <- predict(bagging(spam~., data = x.train), newdata = x.test,
                           type = "class")
         Bagging <- mean(y_pred == x.test$spam)

         #Random forest
         y_pred <- predict(randomForest(spam~., data = x.train), newdata = x.test,
                           type = "class")
         RandomForest <- mean(y_pred == x.test$spam)

         #LDA
         y_pred <- predict(lda(spam~., data = x.train), newdata = x.test,
                           type = "class")$class
         LDA <- mean(y_pred == x.test$spam)

         #QDA
         # Error in qda.default(x, grouping, ...) : rank deficiency in group spam
         #y_pred <- predict(qda(spam~., data = x.train), newdata = x.test)
         #QDA <- mean(y_pred == x.test$spam)

         #Logistic regression
         y_pred <- predict(glm(spam~., family = binomial(), data = x.train),
                           newdata = x.test, type="response")
         LogReg <- mean(y_pred == x.test$spam)

         #Bayes
         y_pred <- predict(naiveBayes(spam~., data = x.train), newdata = x.test)
         Bayes <- mean(y_pred == x.test$spam)

         tibble(Bayes=Bayes, CART=Tree, LDA=LDA, LogReg=LogReg,
                Bagging=Bagging, RandomForest=RandomForest)

       }
      )
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Résultats

```r
#Affichage des résultats sous forme de boxplots
performance <- gather(performance, key="Algorithm", value="Error")
ggplot(data = performance,aes(x=Algorithm, y=Error,col=Algorithm)) + geom_boxplot()
```



Conclusion

Nous remarquons, en comparant les résultats des 7 différentes machines, que l'algorithme *Random Forest* est le plus performant.