projet

2023-04-11

## Importation des données

data <- read.csv("data/Train\_data.csv")

## Preparation du dataset

#transform categorical variable into factor  
data$class<-as.factor(data$class)  
data$protocol\_type<-as.factor(data$protocol\_type)  
data$flag<-as.factor(data$flag)  
#data$service<-as.factor(data$service)  
  
#faire doublons et outliers

table(data$land) #refere to if the connection comes from the intranet

##   
## 0 1   
## 25190 2

table(data$class[data$land == 1])

##   
## anomaly normal   
## 1 1

#The hypotheses of the Fisher's exact test are the same than for the Chi-square test, that is: H0 : the variables are independent, there is no relationship between the two categorical variables.  
tab.cont.land <- xtabs(~class+land,data=data)  
resultat.fisher.land <- fisher.test(tab.cont.land)  
resultat.fisher.land

##   
## Fisher's Exact Test for Count Data  
##   
## data: tab.cont.land  
## p-value = 1  
## alternative hypothesis: true odds ratio is not equal to 1  
## 95 percent confidence interval:  
## 0.01112986 68.51054686  
## sample estimates:  
## odds ratio   
## 0.8731467

table(data$num\_shells)

##   
## 0 1   
## 25183 9

table(data$class[data$num\_shells == 1])

##   
## anomaly normal   
## 1 8

tab.cont.num\_shells <- xtabs(~class+num\_shells,data=data)  
resultat.fisher.num\_shells <- fisher.test(tab.cont.num\_shells)  
resultat.fisher.num\_shells

##   
## Fisher's Exact Test for Count Data  
##   
## data: tab.cont.num\_shells  
## p-value = 0.04292  
## alternative hypothesis: true odds ratio is not equal to 1  
## 95 percent confidence interval:  
## 0.9366956 309.7142399  
## sample estimates:  
## odds ratio   
## 6.988335

table(data$num\_outbound\_cmds)

##   
## 0   
## 25192

table(data$is\_host\_login)

##   
## 0   
## 25192

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

La taille des échantillons sont petites nous allons donc utiliser le test exact de fisher.

Pour land la p-value est supérieur à 0.05 donc nous conservons H0.On considère que land ne dépend pas de class. Nous allons donc retier la variable du dataset.

Pour num\_shells la p-value est inférieur au seil de 5% nous pouvons donc rejeter H0 et considérons que num\_shells depend de class.

Les variables num\_outbound\_cmds et is\_host\_login sont constantes nous pouvons donc les retier.

data <- subset(data, select = -land)  
data <- subset(data, select = -num\_outbound\_cmds)  
data <- subset(data, select = -is\_host\_login)  
data <- subset(data, select = -num\_shells)

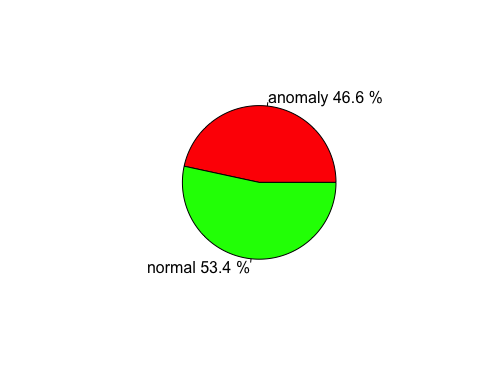
length(unique(data$service))

## [1] 66

#because service has 62 categories and random forest supports up to 53 we tranform the variable into numerical   
#we will do label encoding  
data$service <- as.numeric(factor(data$service))  
  
  
#essayer de mettre 66 categories

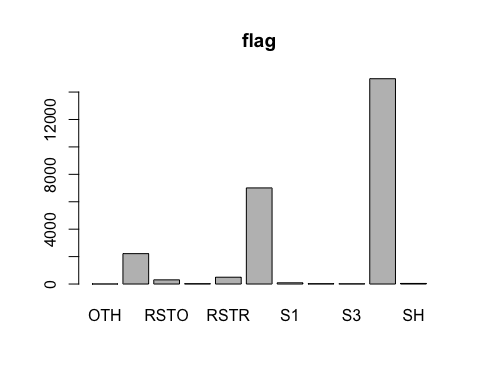
# Analyse descriptive

mytable <- table(data[,"class"])  
prop\_table <- prop.table(mytable)  
label\_percentages <- paste(names(prop\_table), round(prop\_table \* 100, 1), "%", sep = " ")  
pie(mytable, labels = label\_percentages, col = c("red","green"))

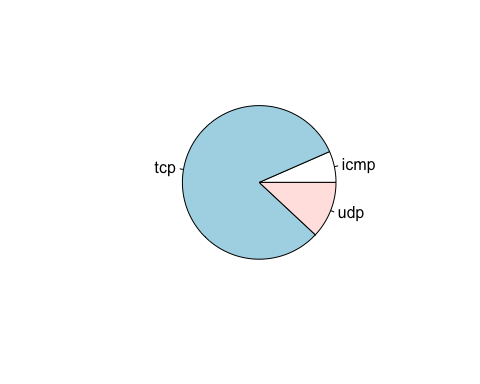


g

barplot(table(data[,"flag"]),main="flag")



pie(table(data[,"protocol\_type"]))



Nous séparons le dataset en 2 pour avoir une partie pour entrainer les modèles et une autre pour les tester

set.seed(5678)  
perm <- sample(4601,3000)  
app <- data[perm,]  
valid <- data[-perm,]

# LDA

library(MASS)

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

model.lda <- lda(class~.,data=app)

## Prédiction

pred\_lda <- predict(model.lda,newdata=valid)$class

## Accuracy

accuracy.lda = mean(pred\_lda==valid$class)  
accuracy.lda

## [1] 0.9534517

## aire sous courbe ROC

library("pROC")

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

pred\_numeric\_lda <- predict(model.lda,newdata=valid)$posterior[,2]  
auc.lda <- roc(valid$class, pred\_numeric\_lda)$auc

## Setting levels: control = anomaly, case = normal

## Setting direction: controls < cases

auc.lda

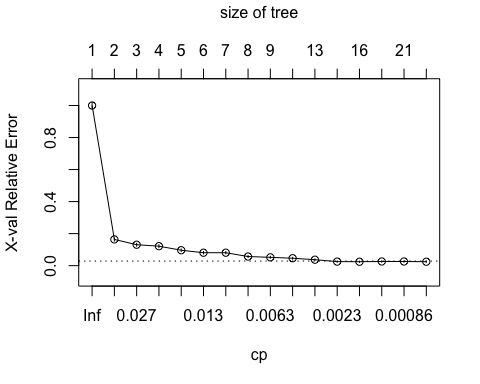
## Area under the curve: 0.9879

# Arbre Optimal

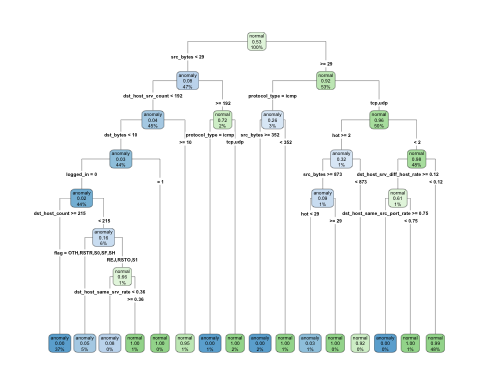
library(rpart)  
library(rpart.plot)  
set.seed(1)  
arbre <- rpart(class~. ,app,control=rpart.control(minsplit=5,cp=0))  
printcp(arbre)

##   
## Classification tree:  
## rpart(formula = class ~ ., data = app, control = rpart.control(minsplit = 5,   
## cp = 0))  
##   
## Variables actually used in tree construction:  
## [1] dst\_bytes dst\_host\_count   
## [3] dst\_host\_diff\_srv\_rate dst\_host\_same\_src\_port\_rate  
## [5] dst\_host\_same\_srv\_rate dst\_host\_srv\_count   
## [7] dst\_host\_srv\_diff\_host\_rate flag   
## [9] hot logged\_in   
## [11] num\_compromised protocol\_type   
## [13] rerror\_rate service   
## [15] src\_bytes srv\_count   
##   
## Root node error: 1422/3000 = 0.474  
##   
## n= 3000   
##   
## CP nsplit rel error xerror xstd  
## 1 0.83614627 0 1.0000000 1.000000 0.0192328  
## 2 0.03305204 1 0.1638537 0.163854 0.0103091  
## 3 0.02180028 2 0.1308017 0.130802 0.0092888  
## 4 0.01758087 3 0.1090014 0.121660 0.0089790  
## 5 0.01336146 4 0.0914205 0.096343 0.0080410  
## 6 0.01265823 5 0.0780591 0.080872 0.0073954  
## 7 0.01125176 6 0.0654008 0.080872 0.0073954  
## 8 0.00703235 7 0.0541491 0.056962 0.0062431  
## 9 0.00562588 8 0.0471167 0.052039 0.0059744  
## 10 0.00515706 9 0.0414909 0.046414 0.0056499  
## 11 0.00386779 12 0.0260197 0.037271 0.0050742  
## 12 0.00140647 14 0.0182841 0.025316 0.0041940  
## 13 0.00117206 15 0.0168776 0.024613 0.0041361  
## 14 0.00105485 18 0.0133615 0.026020 0.0042512  
## 15 0.00070323 20 0.0112518 0.026020 0.0042512  
## 16 0.00000000 26 0.0070323 0.024613 0.0041361

plotcp(arbre)



cp.opt <- arbre$cptable[which.min(arbre$cptable[, "xerror"]), "CP"]   
arbre.opt <- prune(arbre,cp.opt)   
rpart.plot(arbre.opt, type=4)

 ## Prediction

pred\_arbre <- predict(arbre.opt,newdata=valid, type="class")

## Accuracy

accuracy.arbre = mean(pred\_arbre==valid$class)  
accuracy.arbre

## [1] 0.9840934

## aire sous courbe ROC

library("pROC")  
pred\_numeric <- predict(arbre.opt,newdata=valid)[,2]  
auc.rf <- roc(valid$class, pred\_numeric)$auc

## Setting levels: control = anomaly, case = normal

## Setting direction: controls < cases

auc.rf

## Area under the curve: 0.989

# random forest

library("randomForest")

## randomForest 4.7-1.1

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

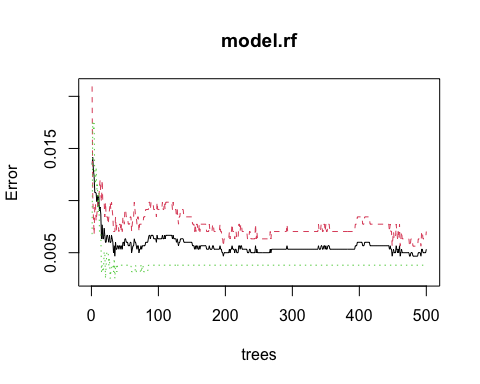
##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

set.seed(1234)  
  
  
model.rf <- randomForest(class~.,data=app)  
model.rf

##   
## Call:  
## randomForest(formula = class ~ ., data = app)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 6  
##   
## OOB estimate of error rate: 0.53%  
## Confusion matrix:  
## anomaly normal class.error  
## anomaly 1412 10 0.007032349  
## normal 6 1572 0.003802281

plot(model.rf)



tail(model.rf$err.rate)

## OOB anomaly normal  
## [495,] 0.005000000 0.006329114 0.003802281  
## [496,] 0.005000000 0.006329114 0.003802281  
## [497,] 0.005000000 0.006329114 0.003802281  
## [498,] 0.005000000 0.006329114 0.003802281  
## [499,] 0.005000000 0.006329114 0.003802281  
## [500,] 0.005333333 0.007032349 0.003802281

## Prediction

pred\_forest <- predict(model.rf,newdata=valid, type="class")

table(pred\_forest,valid$class)

##   
## pred\_forest anomaly normal  
## anomaly 10228 36  
## normal 93 11835

## Accuracy

accuracy.arbre = mean(pred\_forest==valid$class)  
accuracy.arbre

## [1] 0.9941871

## validation

library("pROC")  
  
pred\_numeric <- predict(model.rf, valid, type="prob")[,2]  
auc.rf <- roc(valid$class, pred\_numeric)$auc

## Setting levels: control = anomaly, case = normal

## Setting direction: controls < cases

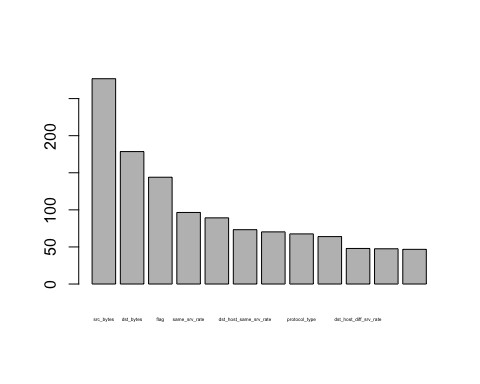
auc.rf

## Area under the curve: 0.9996

var.imp<-model.rf$importance  
var.imp

## MeanDecreaseGini  
## duration 4.19562980  
## protocol\_type 67.57030717  
## service 30.90960822  
## flag 143.98623987  
## src\_bytes 276.78509928  
## dst\_bytes 178.55971814  
## wrong\_fragment 7.07068425  
## urgent 0.26898283  
## hot 13.80692704  
## num\_failed\_logins 0.25518505  
## logged\_in 63.94905336  
## num\_compromised 9.25838910  
## root\_shell 0.17807383  
## su\_attempted 0.03134332  
## num\_root 0.77980782  
## num\_file\_creations 0.12310791  
## num\_access\_files 0.06487435  
## is\_guest\_login 2.58210077  
## count 47.52026263  
## srv\_count 27.00091358  
## serror\_rate 26.76467887  
## srv\_serror\_rate 19.01512246  
## rerror\_rate 7.50113794  
## srv\_rerror\_rate 4.95892316  
## same\_srv\_rate 96.47857112  
## diff\_srv\_rate 89.21001951  
## srv\_diff\_host\_rate 5.30625189  
## dst\_host\_count 24.93262951  
## dst\_host\_srv\_count 70.30518696  
## dst\_host\_same\_srv\_rate 73.25548682  
## dst\_host\_diff\_srv\_rate 47.94906049  
## dst\_host\_same\_src\_port\_rate 46.75897982  
## dst\_host\_srv\_diff\_host\_rate 28.30024786  
## dst\_host\_serror\_rate 20.11848459  
## dst\_host\_srv\_serror\_rate 27.42050759  
## dst\_host\_rerror\_rate 14.39690752  
## dst\_host\_srv\_rerror\_rate 11.47170989

ord <- order(var.imp,decreasing = TRUE)  
barplot(sort(var.imp,decreasing = TRUE)[1:12], names.arg=rownames(var.imp)[ord][1:12],cex.names=0.3)



var.imp <- as.data.frame(var.imp)  
var.imp.df <- cbind(variables = rownames(var.imp),var.imp)  
var.imp.df <- var.imp[order(var.imp$MeanDecreaseGini, decreasing = TRUE),]  
rownames(var.imp.df) <- NULL  
head(var.imp.df,10)

## [1] 276.78510 178.55972 143.98624 96.47857 89.21002 73.25549 70.30519  
## [8] 67.57031 63.94905 47.94906

### src\_bytes number of data bytes from source to destination

# gradient boosting

library(gbm)

## Loaded gbm 2.1.8.1

app2<-app  
valid2 <- valid  
app2$service<-as.factor(app2$service)  
valid2$service<-as.factor(valid2$service)  
  
app2$class <- ifelse(app2$class == "normal", 1, 0)  
valid2$class <- ifelse(valid2$class == "normal", 1, 0)

mod.ada<-gbm(class~., data=app2, distribution="adaboost", shrinkage=0.01,n.trees=3000,cv.folds = 5)

print(mod.ada)

## gbm(formula = class ~ ., distribution = "adaboost", data = app2,   
## n.trees = 3000, shrinkage = 0.01, cv.folds = 5)  
## A gradient boosted model with adaboost loss function.  
## 3000 iterations were performed.  
## The best cross-validation iteration was 2982.  
## There were 37 predictors of which 25 had non-zero influence.

library("pROC")  
prev.ada <- predict(mod.ada,newdata=valid,type = "response")

## Using 2982 trees...

prev.ada <- round(prev.ada)  
auc.ada <- roc(valid2$class, prev.ada)$auc

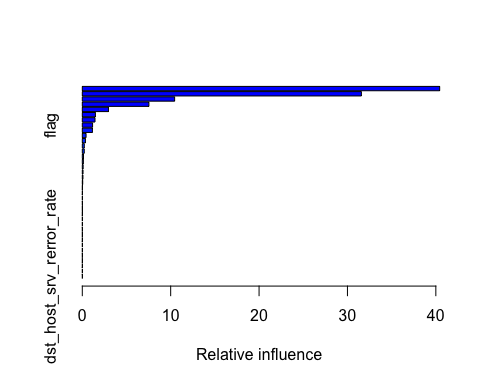
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

auc.ada

## Area under the curve: 0.9135

summary(mod.ada)[1:10,]



## var rel.inf  
## service service 40.4352350  
## src\_bytes src\_bytes 31.5760946  
## dst\_bytes dst\_bytes 10.4478351  
## num\_compromised num\_compromised 7.5372878  
## hot hot 2.9600049  
## same\_srv\_rate same\_srv\_rate 1.4825520  
## count count 1.4376703  
## flag flag 1.1634194  
## dst\_host\_same\_src\_port\_rate dst\_host\_same\_src\_port\_rate 1.1422605  
## diff\_srv\_rate diff\_srv\_rate 0.4265565

# regression Logistique

mod.reg <- glm(class ~ ., family = binomial , data=app2)

## Warning: glm.fit: algorithm did not converge

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

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-7

lasso <- glmnet(as.matrix(app[,1:37]),app[,38],family="binomial",alpha=0)

## Warning in storage.mode(xd) <- "double": NAs introduced by coercion

#model.log.asso <- glmnet(as.matrix(app[,-1]),app$class,family='binomial')  
#cvLasso <- cv.glmnet(as.matrix(app[,-1]),app$class,family="binomial", type.measure = "class")  
#plot(cvLasso)