



HEC MONTRÉAL

Projet MATH60603

Prévisions de perte de clientèle

DAVID GARSON, 11262368
JEAN GUIRGUIS, NUM-MATRICULE

Projet présenté à

PROFESSEURE AURÉLIE LABBÉ

Table des mati  res

1 Introduction : pr��sentation des objectifs de l��tude	2
2 M��thode : pr��sentation du design de l��tude et des diff��rents sc��narios utilis��s. Pr��sentation de la m��thode d��analyse en g��n��ral.	2
2.1 Exploration des donn��es	2
3 Exploration des donn��es	2
3.1 R��encodage des variables “State”, “International.plan” et “Voice.mail.plan” en facteur. La variable d��int��r��t est r��encod��e en variable logique, �� savoir que les valeurs “vrai” sont cod��es “1” et les valeurs “faux” sont cod��es en 0	4
3.2 ��tude descriptive des donn��es	4
3.3 Analyse descriptive des donn��es	7
3.4 R��gression logistique (validation crois��) — a corriger la courbe roc	12
3.5 Arbre de classification ��lag��	15
3.6 for��t al��atoire approche bagging	19
3.7 Boosting	22
3.8 SVM	23
4 R��sultats : pr��sentation des r��sultats sous forme de tableaux et figures (ne mettez pas de sortie R)	23
5 Conclusion/discussion : conclusion g��n��rale, limites de votre ��tude, qu��avez vous appris ?	23

Note :

baggin + svm + regression linéaire

équilibrer VS non équilibré (3 méthode)

1. Introduction : présentation des objectifs de l'étude

L'objectif de cette étude est de faire une prédiction sur le désabonnement d'un client pour un service de télécommunication. Cette simulation a été faite à partir d'une base de données présente sur kaggle à l'adresse suivante : <https://www.kaggle.com/mnassrib/telecom-churn-datasets?select=churn-bigml-80.csv>. L'analyse a été effectuée sur les clients d'Orange Telecom aux États-Unis et plusieurs variables ont été analysées tels que lieu géographique, le type de plan que le client possède, le nombre de minutes utilisé pour des appels durant le jour et le soir, etc.

Ce rapport contient en premier lieu, une présentation détaillée de notre méthode d'analyse. Cette section contiendra d'abord une exploration détaillée des données possédées, un traitement et un nettoyage préparatoire des données. Puis, plusieurs méthodes et stratégies de prédiction seront testées afin de déterminer le modèle le mieux adapté à notre échantillon d'utilisateurs.

Une fois le modèle choisi, la deuxième section du rapport présentera les résultats du modèle de prédiction.

... ..

2. Méthode : présentation du design de l'étude et des différents scénarios utilisés. Présentation de la méthode d'analyse en général.

2.1. Exploration des données

2.1.1 Présentation des variables

2.1.2 Traitement et nettoyage préparatoire des données

3. Exploration des données

```
# Importation des données

train0=read.csv("data/churn-bigml-80.csv")
nb_ligne_train = nrow(train0)
test0=read.csv("data/churn-bigml-20.csv")
mydata = rbind(train0,test0)
id.train=c(1:nrow(train0))
id.test=c((nrow(train0)+1):(nrow(mydata)))

train= mydata[id.train,]
test = mydata[id.test,]
attributes(test)$row.names = attributes(test0)$row.names
identical(train,train0)

## [1] TRUE
```

```
identical(test,test0)
```

```
## [1] TRUE
```

```
all.equal(train,train0)
```

```
## [1] TRUE
```

```
all.equal(test,test0)
```

```
## [1] TRUE
```

```
summary(train)
```

```
##      State      Account.length      Area.code      International.plan
## Length:2666      Min.   : 1.0      Min.   :408.0      Length:2666
## Class :character 1st Qu.: 73.0      1st Qu.:408.0      Class :character
## Mode  :character Median :100.0      Median :415.0      Mode  :character
##                      Mean  :100.6      Mean  :437.4
##                      3rd Qu.:127.0      3rd Qu.:510.0
##                      Max.   :243.0      Max.   :510.0
## Voice.mail.plan  Number.vmail.messages Total.day.minutes Total.day.calls
## Length:2666      Min.   : 0.000      Min.   : 0.0      Min.   : 0.0
## Class :character 1st Qu.: 0.000      1st Qu.:143.4      1st Qu.: 87.0
## Mode  :character Median : 0.000      Median :179.9      Median :101.0
##                      Mean   : 8.022      Mean   :179.5      Mean   :100.3
##                      3rd Qu.:19.000      3rd Qu.:215.9      3rd Qu.:114.0
##                      Max.    :50.000      Max.    :350.8      Max.    :160.0
## Total.day.charge Total.eve.minutes Total.eve.calls Total.eve.charge
## Min.   : 0.00      Min.   : 0.0      Min.   : 0      Min.   : 0.00
## 1st Qu.:24.38      1st Qu.:165.3      1st Qu.: 87      1st Qu.:14.05
## Median :30.59      Median :200.9      Median :100      Median :17.08
## Mean   :30.51      Mean   :200.4      Mean   :100      Mean   :17.03
## 3rd Qu.:36.70      3rd Qu.:235.1      3rd Qu.:114      3rd Qu.:19.98
## Max.    :59.64      Max.    :363.7      Max.    :170      Max.    :30.91
## Total.night.minutes Total.night.calls Total.night.charge Total.intl.minutes
## Min.   : 43.7      Min.   : 33.0      Min.   : 1.970      Min.   : 0.00
## 1st Qu.:166.9      1st Qu.: 87.0      1st Qu.: 7.513      1st Qu.: 8.50
## Median :201.2      Median :100.0      Median : 9.050      Median :10.20
## Mean   :201.2      Mean   :100.1      Mean   : 9.053      Mean   :10.24
## 3rd Qu.:236.5      3rd Qu.:113.0      3rd Qu.:10.640      3rd Qu.:12.10
## Max.    :395.0      Max.    :166.0      Max.    :17.770      Max.    :20.00
## Total.intl.calls Total.intl.charge Customer.service.calls Churn
## Min.   : 0.000      Min.   :0.000      Min.   :0.000      Length:2666
## 1st Qu.: 3.000      1st Qu.:2.300      1st Qu.:1.000      Class :character
## Median : 4.000      Median :2.750      Median :1.000      Mode  :character
## Mean   : 4.467      Mean   :2.764      Mean   :1.563
## 3rd Qu.: 6.000      3rd Qu.:3.270      3rd Qu.:2.000
## Max.    :20.000      Max.    :5.400      Max.    :9.000
```

3.1. R  encodage des variables “State”, “International.plan” et “Voice.mail.plan” en facteur. La variable d’int  r  t est r  encod  e en variable logique,    savoir que les valeurs “vrai” sont cod  es “1” et les valeurs “faux” sont cod  es en 0

```
#transformation des variables "character" en "facteur"

mydata$State=as.factor(mydata$State)
mydata$International.plan=as.factor(mydata$International.plan)
mydata$Voice.mail.plan=as.factor(mydata$Voice.mail.plan)

#transformation variable d'int  r  t en variable logique
mydata$Churn=as.logical(mydata$Churn)

train= mydata[id.train,]
test = mydata[id.test,]
attributes(test)$row.names = attributes(test0)$row.names
```

3.2.   tude descriptive des donn  es

```
# fonction
analyse_table = function (nom,variable,nb_donne)
{
  table_temporaire= table(variable)
  table_temporaire = as.data.frame(table_temporaire)
  table_temporaire = data.frame(table_temporaire,pourcentage=round(table_temporaire[2]/nb_donne
names(table_temporaire)[3] = "% Freq"
names(table_temporaire)[1] = nom
#table_temporaire = head(table_temporaire[order(-table_temporaire[3]),],3)
  table_temporaire = table_temporaire[order(-table_temporaire[3]),]
  return(table_temporaire)
}

analyse_table('State',train$State,nrow(train))
```

```
##      State Freq % Freq
## 50     WV   88  3.30
## 24     MN   70  2.63
## 35     NY   68  2.55
## 46     VA   67  2.51
## 2      AL   66  2.48
## 36     OH   66  2.48
## 51     WY   66  2.48
## 38     OR   62  2.33
## 34     NV   61  2.29
## 49     WI   61  2.29
## 21     MD   60  2.25
## 45     UT   60  2.25
## 6      CO   59  2.21
## 7      CT   59  2.21
## 23     MI   58  2.18
## 47     VT   57  2.14
```

```
## 14    ID    56    2.10
## 28    NC    56    2.10
## 44    TX    55    2.06
## 10    FL    54    2.03
## 16    IN    54    2.03
## 27    MT    53    1.99
## 17    KS    52    1.95
## 20    MA    52    1.95
## 37    OK    52    1.95
## 9     DE    51    1.91
## 25    MO    51    1.91
## 32    NJ    50    1.88
## 11    GA    49    1.84
## 22    ME    49    1.84
## 41    SC    49    1.84
## 42    SD    49    1.84
## 26    MS    48    1.80
## 40    RI    48    1.80
## 48    WA    48    1.80
## 3     AR    47    1.76
## 4     AZ    45    1.69
## 8     DC    45    1.69
## 15    IL    45    1.69
## 30    NE    45    1.69
## 12    HI    44    1.65
## 29    ND    44    1.65
## 33    NM    44    1.65
## 1     AK    43    1.61
## 18    KY    43    1.61
## 31    NH    43    1.61
## 43    TN    41    1.54
## 13    IA    38    1.43
## 39    PA    36    1.35
## 19    LA    35    1.31
## 5     CA    24    0.90
```

```
analyse_table('International.plan',train$International.plan,nrow(train))
```

```
##      International.plan Freq % Freq
## 1                      No 2396  89.87
## 2                      Yes  270  10.13
```

```
analyse_table('Voice.mail.plan',train$Voice.mail.plan,nrow(train))
```

```
##      Voice.mail.plan Freq % Freq
## 1                   No 1933  72.51
## 2                   Yes  733  27.49
```

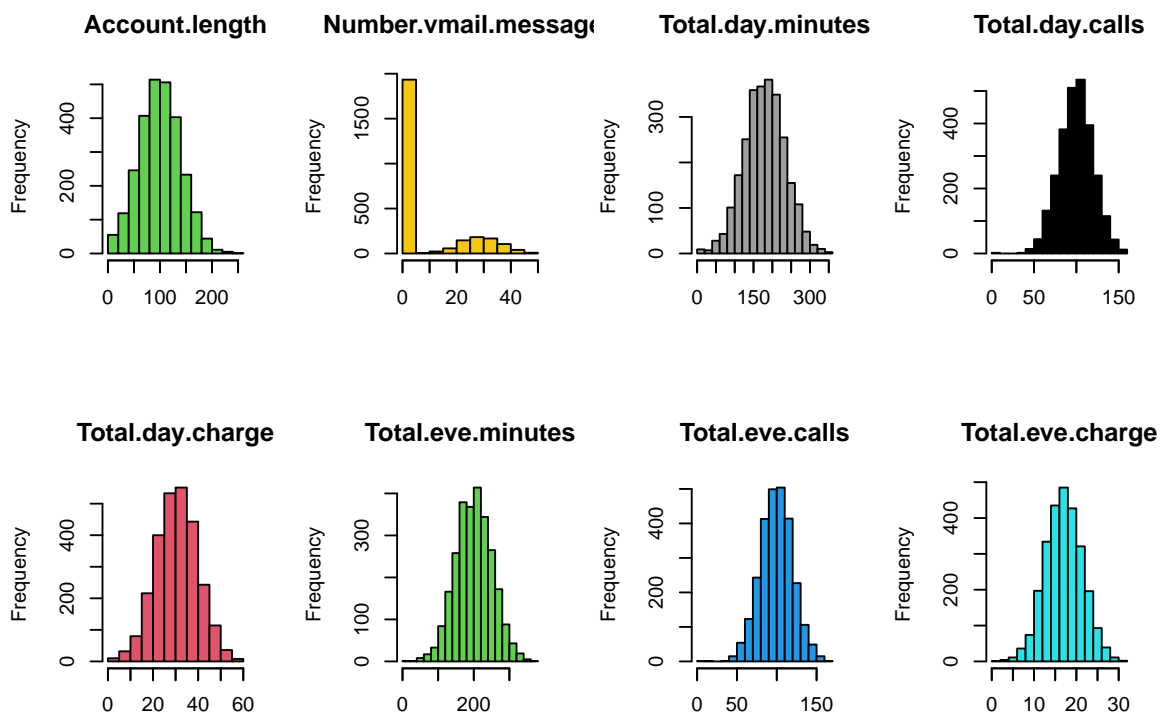
```
analyse_table('Churn',train$Churn,nrow(train))
```

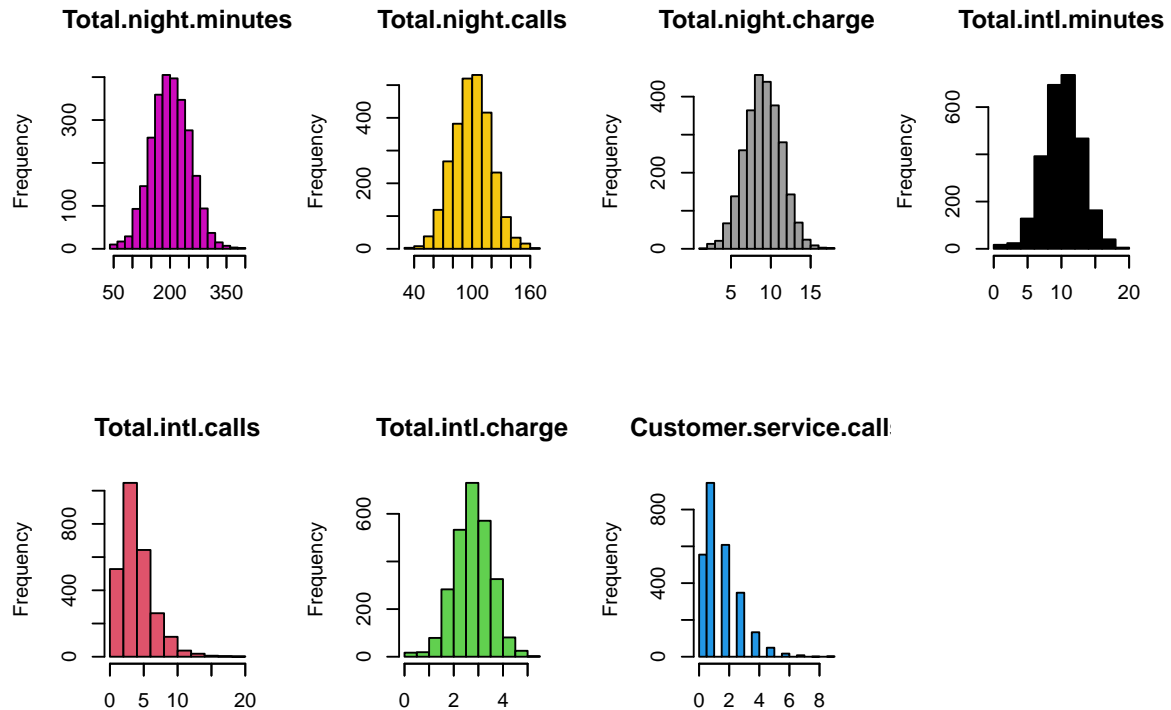
```
##      Churn Freq % Freq
## 1 FALSE 2278  85.45
## 2  TRUE  388  14.55
```

```
analyse_table('Area.code',train$Area.code,nrow(train))
```

```
##   Area.code Freq % Freq
## 2      415 1318 49.44
## 3      510  679 25.47
## 1      408  669 25.09
```

```
par(mfrow=c(2,4))
for (i in 1:length(colnames(train)))
{
  if (i != 1 & i != 3 & i != 4 & i != 5 & i !=length(colnames(train)))
  {
    hist(train[,i], main = (colnames(train)[i] ),xlab = element_blank(),col = i+1)
    Axis(side=1, labels=FALSE)
    #plot(train[,i], main = (colnames(train)[i] ))
  }
}
```





REMARQUE : Si on conserve les variables Customer.service.calls et total.intl.calls en variables numériques, il faudra sûrement prendre le log de ces variables car elles sont asymétriques à droite!!

3.3. Analyse descriptive des données

```
summary(train)
```

```
##      State      Account.length      Area.code      International.plan
## WV       : 88      Min.       : 1.0      Min.       :408.0      No :2396
## MN       : 70      1st Qu.: 73.0      1st Qu.:408.0      Yes: 270
## NY       : 68      Median :100.0      Median :415.0
## VA       : 67      Mean    :100.6      Mean    :437.4
## AL       : 66      3rd Qu.:127.0      3rd Qu.:510.0
## OH       : 66      Max.     :243.0      Max.     :510.0
## (Other):2241
## Voice.mail.plan Number.vmail.messages Total.day.minutes Total.day.calls
## No :1933      Min.       : 0.000      Min.       : 0.0      Min.       : 0.0
## Yes: 733      1st Qu.: 0.000      1st Qu.:143.4      1st Qu.: 87.0
##              Median : 0.000      Median :179.9      Median :101.0
##              Mean    : 8.022      Mean    :179.5      Mean    :100.3
##              3rd Qu.:19.000      3rd Qu.:215.9      3rd Qu.:114.0
##              Max.     :50.000      Max.     :350.8      Max.     :160.0
##
## Total.day.charge Total.eve.minutes Total.eve.calls Total.eve.charge
## Min.       : 0.00      Min.       : 0.0      Min.       : 0      Min.       : 0.00
## 1st Qu.:24.38      1st Qu.:165.3      1st Qu.: 87      1st Qu.:14.05
## Median :30.59      Median :200.9      Median :100      Median :17.08
```



```
## Mean :30.51 Mean :200.4 Mean :100 Mean :17.03
## 3rd Qu.:36.70 3rd Qu.:235.1 3rd Qu.:114 3rd Qu.:19.98
## Max. :59.64 Max. :363.7 Max. :170 Max. :30.91
##
## Total.night.minutes Total.night.calls Total.night.charge Total.intl.minutes
## Min. : 43.7 Min. : 33.0 Min. : 1.970 Min. : 0.00
## 1st Qu.:166.9 1st Qu.: 87.0 1st Qu.: 7.513 1st Qu.: 8.50
## Median :201.2 Median :100.0 Median : 9.050 Median :10.20
## Mean :201.2 Mean :100.1 Mean : 9.053 Mean :10.24
## 3rd Qu.:236.5 3rd Qu.:113.0 3rd Qu.:10.640 3rd Qu.:12.10
## Max. :395.0 Max. :166.0 Max. :17.770 Max. :20.00
##
## Total.intl.calls Total.intl.charge Customer.service.calls Churn
## Min. : 0.000 Min. :0.000 Min. :0.000 Mode :logical
## 1st Qu.: 3.000 1st Qu.:2.300 1st Qu.:1.000 FALSE:2278
## Median : 4.000 Median :2.750 Median :1.000 TRUE :388
## Mean : 4.467 Mean :2.764 Mean :1.563
## 3rd Qu.: 6.000 3rd Qu.:3.270 3rd Qu.:2.000
## Max. :20.000 Max. :5.400 Max. :9.000
##
```

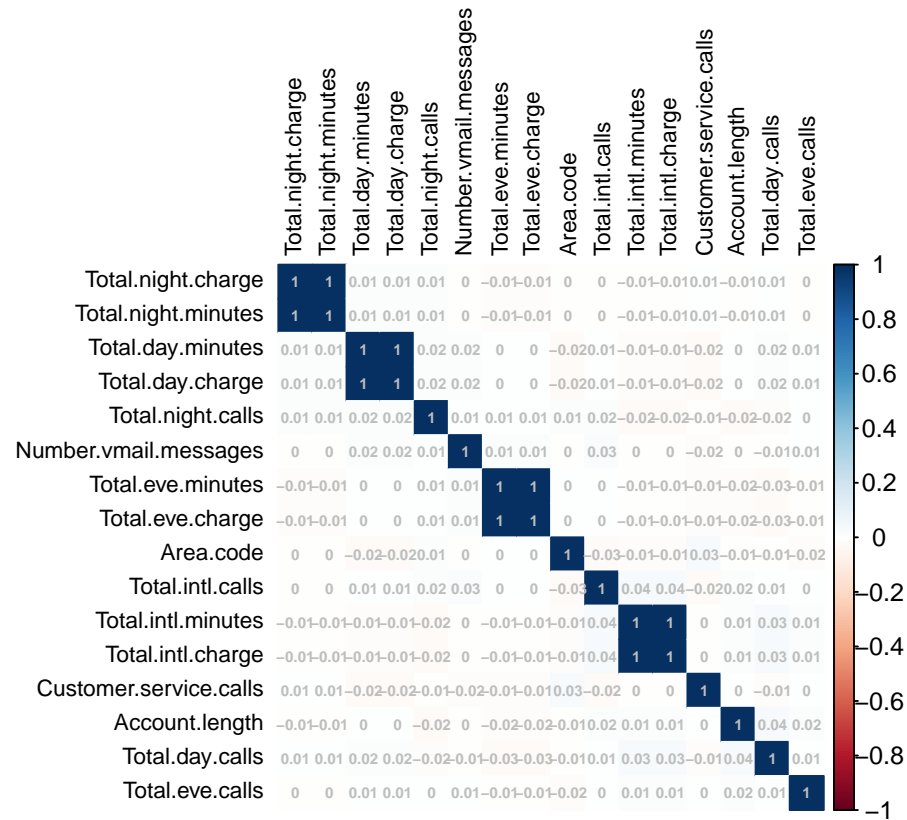
Le dataset d'entra nement ne contient aucune valeur manquante. il y a 15 variables continues, 3 variables cat gorielles, et une variables binaires. (a faire avec cours chapitre 9 is na)

```
round(mean(train$Churn),digits=2)
```

```
## [1] 0.15
```

Le pourcentage de clients ayant quitt s la compagnie est de 15%.

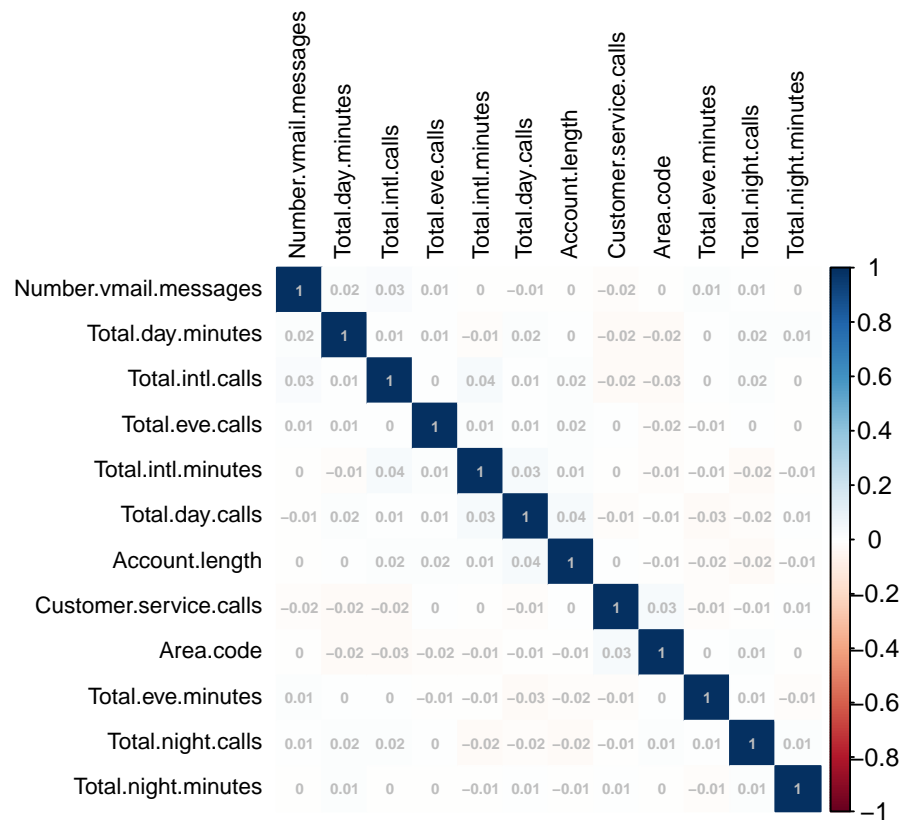
```
corrplot(cor(train[,c(2,3,6:19)]), method = "color", addCoef.col="grey", order = "AOE",tl.cex=0
```



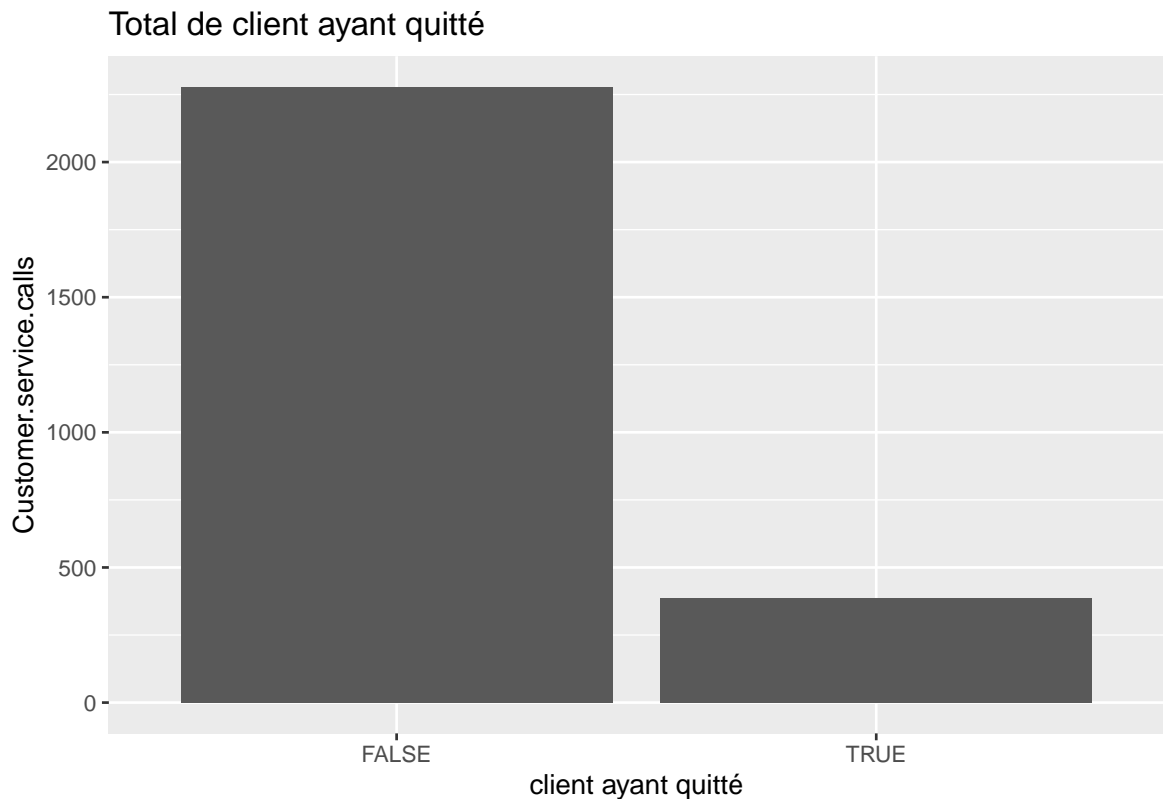
Les variables du dataset sont faiblement corr  l  es    l'exception des variables indiquant le nombre de minutes consomm  es et les frais charg  s associ  s, comme les variables : "Total.night charge" et "Total.night.minutes". Comme ces variables ont une corr  lation parfaites, nous d  cisons de supprimer du dataset les variables "charge". Ce qui revient    supprimer 4 variables du dataset.

```
mydata = subset(mydata, select = -c(Total.day.charge,Total.night.charge,Total.eve.charge,Total.
train= mydata[id.train,]
test = mydata[id.test,]
attributes(test)$row.names = attributes(test0)$row.names

corrplot(cor(train[,c(2,3,6:(length(names(mydata))-1))]), method = "color", addCoef.col="grey",
```



```
library(ggplot2)
ggplot(train, aes(x = Churn, fill=Customer.service.calls)) +
  geom_bar( ) +
  xlab("client ayant quitt  ") + ylab("Customer.service.calls") +
  ggtitle("Total de client ayant quitt  ")
```



```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.3
```

```
library(usmap)
```

```
## Warning: package 'usmap' was built under R version 4.0.3
```

```
us.map=train
names(us.map)[names(us.map)=="State"]="state"
us.map = data.frame(us.map)

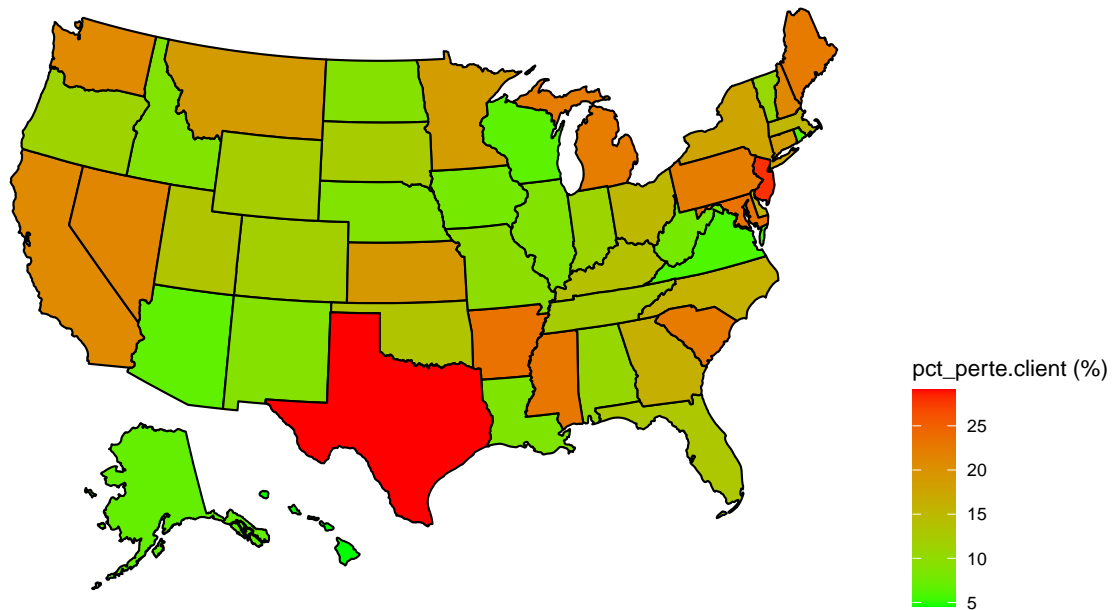
state.churn=us.map %>%
  group_by(state) %>%
  summarise(pct_perte.client = mean(Churn)*100)
```

```
us.map = data.frame(us.map)
```

```
plot_usmap(regions="state", data=state.churn, values = "pct_perte.client", color="black")+
  scale_fill_continuous(low = "green", high = "red", name = "pct_perte.client (%)")+
  labs(title = "Perte client le  tats-Unis", subtitle = "Op rateur Orange t l com")+
  theme(legend.position = "right")
```

Perte client  le   tats  Unis

Op  rateur Orange t  l  com



On constate que les   tats du Texas et New Jersey sont les   tats ayant perdus le plus de client  le, avec un pourcentage de perte sup  rieur    25%.

3.4. R  gression logistique (validation crois  ) — a corriger la courbe roc

.....

```
train = train0
test = train0

set.seed(1234)

train$Churn[train$Churn == "False"] = 0
train$Churn[train$Churn == "True"] = 1
train$Churn = as.numeric(train$Churn)

test$Churn[test$Churn == "False"] = 0
test$Churn[test$Churn == "True"] = 1
test$Churn = as.numeric(test$Churn)

model1=glm(Churn~ ., family="binomial",data=train)
summary(model1)

##
## Call:
## glm(formula = Churn ~ ., family = "binomial", data = train)
```

```

##
## Deviance Residuals:
##      Min        1Q      Median        3Q        Max
## -1.9878   -0.4968   -0.3075   -0.1580    3.1228
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -8.920e+00  1.260e+00  -7.079 1.45e-12 ***
## StateAL        2.075e-01  7.925e-01   0.262  0.79346
## StateAR        8.758e-01  7.711e-01   1.136  0.25607
## StateAZ        1.406e-01  9.246e-01   0.152  0.87916
## StateCA        1.309e+00  8.860e-01   1.477  0.13969
## StateCO        3.176e-01  7.994e-01   0.397  0.69114
## StateCT        1.150e+00  7.500e-01   1.533  0.12524
## StateDC        7.592e-01  8.254e-01   0.920  0.35770
## StateDE        6.462e-01  7.753e-01   0.833  0.40463
## StateFL        5.189e-01  7.931e-01   0.654  0.51291
## StateGA        7.058e-01  7.902e-01   0.893  0.37175
## StateHI       -7.328e-01  1.002e+00  -0.731  0.46457
## StateIA        3.493e-01  9.146e-01   0.382  0.70252
## StateID        3.175e-01  8.344e-01   0.381  0.70355
## StateIL       -4.100e-01  8.949e-01  -0.458  0.64683
## StateIN        2.688e-01  8.043e-01   0.334  0.73824
## StateKS        9.024e-01  7.661e-01   1.178  0.23883
## StateKY        5.636e-01  8.182e-01   0.689  0.49096
## StateLA        5.682e-01  9.068e-01   0.627  0.53091
## StateMA        1.069e+00  7.853e-01   1.361  0.17346
## StateMD        9.575e-01  7.461e-01   1.283  0.19937
## StateME        1.375e+00  7.587e-01   1.813  0.06987 .
## StateMI        1.285e+00  7.462e-01   1.722  0.08506 .
## StateMN        1.175e+00  7.426e-01   1.583  0.11353
## StateMO        3.961e-01  8.306e-01   0.477  0.63344
## StateMS        1.505e+00  7.654e-01   1.967  0.04920 *
## StateMT        1.645e+00  7.562e-01   2.176  0.02957 *
## StateNC        2.904e-01  7.996e-01   0.363  0.71650
## StateND        2.940e-03  8.741e-01   0.003  0.99732
## StateNE        2.949e-01  8.536e-01   0.345  0.72973
## StateNH        1.305e+00  7.888e-01   1.654  0.09811 .
## StateNJ        1.398e+00  7.471e-01   1.871  0.06134 .
## StateNM        3.212e-01  8.655e-01   0.371  0.71052
## StateNV        1.040e+00  7.450e-01   1.395  0.16289
## StateNY        1.111e+00  7.467e-01   1.488  0.13671
## StateOH        7.216e-01  7.650e-01   0.943  0.34559
## StateOK        5.418e-01  7.982e-01   0.679  0.49728
## StateOR        3.802e-01  7.912e-01   0.481  0.63080
## StatePA        1.244e+00  8.055e-01   1.544  0.12260
## StateRI       -9.942e-01  9.569e-01  -1.039  0.29882
## StateSC        1.726e+00  7.675e-01   2.249  0.02451 *
## StateSD        4.039e-01  8.078e-01   0.500  0.61703
## StateTN        3.497e-01  8.477e-01   0.412  0.67998
## StateTX        1.809e+00  7.353e-01   2.461  0.01386 *
## StateUT        9.522e-01  7.754e-01   1.228  0.21945
## StateVA       -6.646e-01  8.765e-01  -0.758  0.44833
## StateVT       -2.364e-01  8.448e-01  -0.280  0.77962
## StateWA        1.326e+00  7.692e-01   1.724  0.08473 .

```

```
## StateWI          -1.862e-01  8.740e-01  -0.213  0.83128
## StateWV          2.495e-01  7.771e-01   0.321  0.74819
## StateWY          2.085e-01  7.812e-01   0.267  0.78952
## Account.length    1.482e-03  1.630e-03   0.909  0.36343
## Area.code        -3.446e-04  1.532e-03  -0.225  0.82202
## International.planYes 2.278e+00  1.704e-01  13.373 < 2e-16 ***
## Voice.mail.planYes -2.159e+00  6.811e-01  -3.170  0.00152 **
## Number.vmail.messages 4.076e-02  2.143e-02   1.902  0.05721 .
## Total.day.minutes -3.491e-01  3.826e+00  -0.091  0.92729
## Total.day.calls    3.608e-03  3.246e-03   1.111  0.26643
## Total.day.charge   2.132e+00  2.251e+01   0.095  0.92455
## Total.eve.minutes  1.419e+00  1.921e+00   0.738  0.46029
## Total.eve.calls    -1.615e-03  3.220e-03  -0.501  0.61609
## Total.eve.charge   -1.662e+01  2.260e+01  -0.735  0.46223
## Total.night.minutes 4.749e-04  1.023e+00   0.000  0.99963
## Total.night.calls  1.735e-03  3.294e-03   0.527  0.59839
## Total.night.charge  5.718e-02  2.272e+01   0.003  0.99799
## Total.intl.minutes -2.985e+00  6.178e+00  -0.483  0.62895
## Total.intl.calls   -1.228e-01  3.005e-02  -4.086  4.38e-05 ***
## Total.intl.charge  1.141e+01  2.288e+01   0.499  0.61795
## Customer.service.calls 5.463e-01  4.662e-02  11.718 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 2212.2  on 2665  degrees of freedom
## Residual deviance: 1641.2  on 2597  degrees of freedom
## AIC: 1779.2
##
## Number of Fisher Scoring iterations: 6
```

```
#BIC
round(BIC(model1), digit = 2)
```

```
## [1] 2185.52
```

```
#AIC
round(model1$aic, digit = 2)
```

```
## [1] 1779.22
```

```
library(ROCR)
prob.predict=predict.glm(model1,test,type="response")
cutoff=0.5
test.pred = rep(0, nrow(test))
test.pred[prob.predict > cutoff] = 1
M=table(test.pred, test$Churn,dnn=c("Prediction","Observation"))

a=M[1,1]
b=M[1,2]
c=M[2,1]
d=M[2,2]
```

```

# Sensibilit  
Sensibilit   = d/(b+d)

# Sp  cificit  
Sp  cificit   = a/(a+c)

# Taux d'erreurs de classification (%)
((b+c)/(a+b+c+d))*100

```

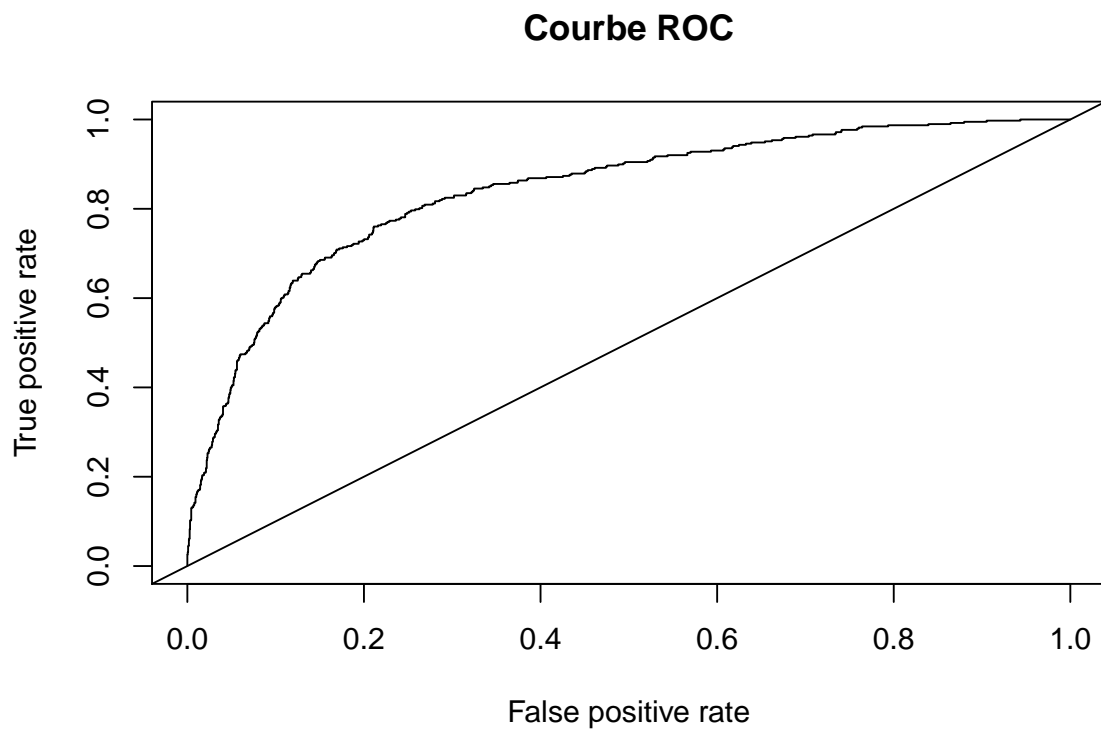
```
## [1] 12.90323
```

```

pred=prediction(prob.predict,test$Churn)
perf=performance(pred,measure="tpr",x.measure="fpr")
auc.perf = performance(pred, measure = "auc")

plot(perf, main = "Courbe ROC")
abline(a=0,b=1)

```



3.5. Arbre de classification   lag  

```

train = train0
test = train0

train$Churn[train$Churn == "False"] = 0
train$Churn[train$Churn == "True"] = 1

```



```

train$Churn = as.numeric(train$Churn)

test$Churn[test$Churn == "False"] = 0
test$Churn[test$Churn == "True"] = 1
test$Churn = as.numeric(test$Churn)

set.seed(400)

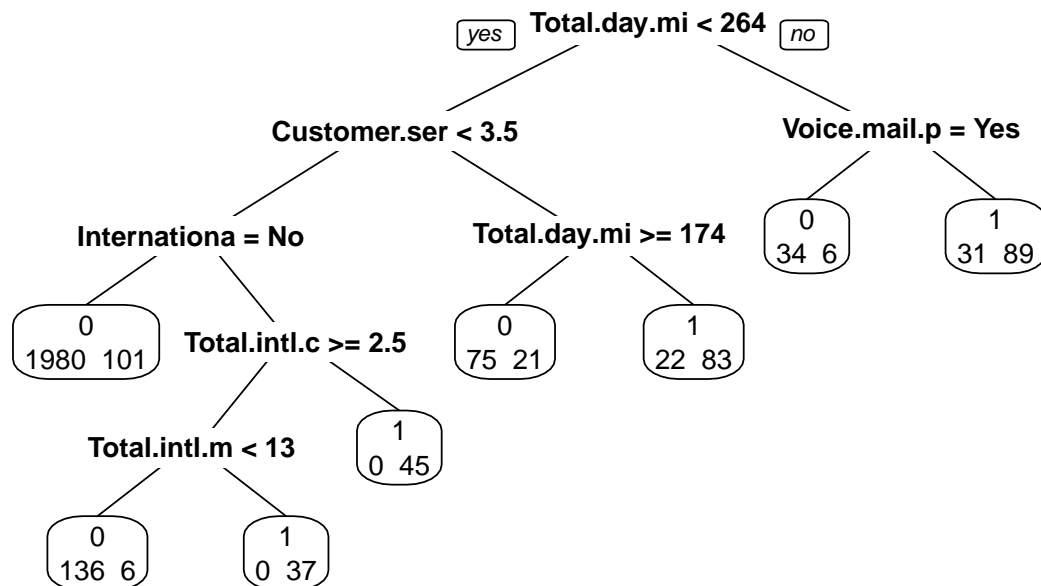
# Cr  ation de l'arbre :
library(rpart.plot)

## Loading required package: rpart

mytree = rpart(Churn~., data=train , method = "class")
cp_optimal=mytree$cptable[which.min(mytree$cptable[,4]),1]
mytree_optimal = prune(mytree,cp=cp_optimal)
prp(mytree_optimal,extra=1,roundint=FALSE, main = "Arbre avec   lagage")

```

Arbre avec   lagage



```

mytable=table(test$Churn, predict(mytree_optimal,test, type="class"))
names(dimnames(mytable))= c("Observed", "Predicted")
M = mytable

a=M[1,1]
b=M[1,2]
c=M[2,1]
d=M[2,2]

```

```
# Taux de mauvaises classifications
((b+c)/(a+b+c+d))*100
```

```
## [1] 7.014254
```

```
# Taux de faux positifs
round(((b+c)/(a+b+c+d))*100, digit = 2)
```

```
## [1] 7.01
```

```
# Taux faux négatif
round((1-(d/(b+d)))*100, digit = 2)
```

```
## [1] 17.26
```

```
# Paramètre de complexité maximale
cp_optimal
```

```
## [1] 0.02835052
```

```
# Boucle for
```

```
nb_boucle = 20
```

```
Taux_mauvaise_classificaion=matrix(0,nb_boucle,1)
Taux_faux_positifs=matrix(0,nb_boucle,1)
Taux_faux_négatif=matrix(0,nb_boucle,1)
valeur_seed = matrix(0,nb_boucle,1)
```

```
par(mfrow=c(4,3))
```

```
for (i in 1:nb_boucle)
{
  n=nrow(mydata)
  set.seed(i*15231)
  id.train=sample(1:n,size=nrow(train))
  id.test=setdiff(1:n,id.train)
```

```
mydata.train= mydata[id.train,]
mydata.test = mydata[id.test,]
```

```
library(rpart.plot)
set.seed(i*15231)
valeur_seed[i]=i*15231
```

```
mytree = rpart(Churn~., data=mydata.train, method = "class")
```

```
cp_optimal=mytree$cptable[which.min(mytree$cptable[,4]),1]
mytree_optimal = prune(mytree,cp=cp_optimal)
```

```
#prp(mytree_optimal,extra=1,roundint=FALSE)
```

```

mytable=table(mydata.test$Churn, predict(mytree_optimal,mydata.test, type="class"))
names(dimnames(mytable))= c("Observed", "Predicted")
M = mytable

a=M[1,1]
b=M[1,2]
c=M[2,1]
d=M[2,2]

# Taux de mauvaises classifications
Taux_mauvaise_classificaion[i]= ((b+c)/(a+b+c+d))
# Taux faux positifs
Taux_faux_positifs[i]= round((1-(a/(a+c))), digit = 2 )
# Taux faux négatif
Taux_faux_negatif[i] = round((1-(d/(b+d))), digit = 2)

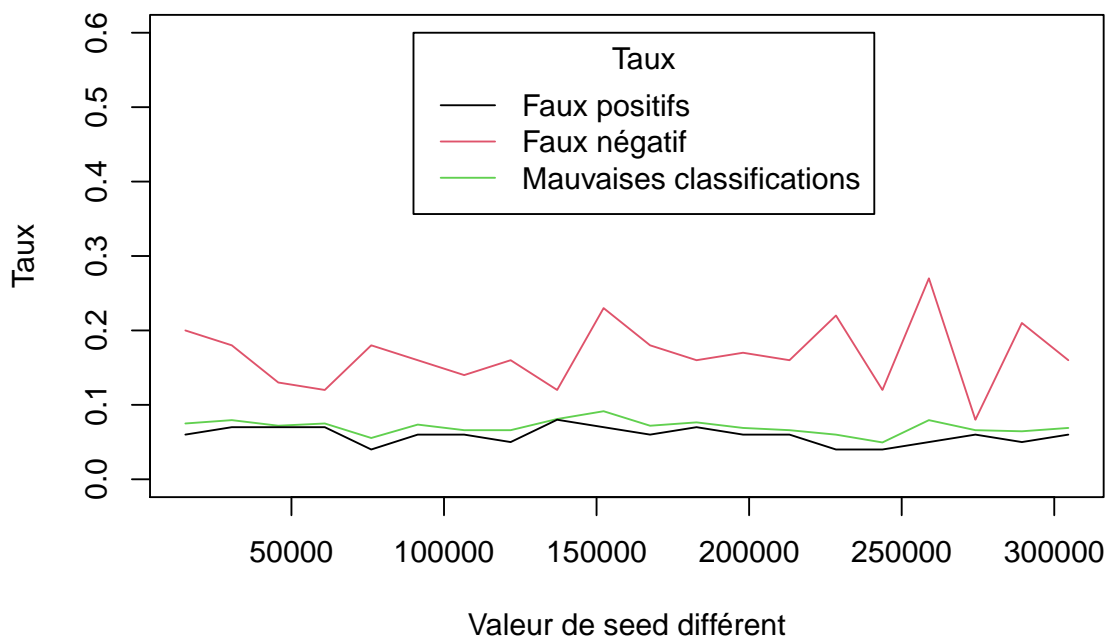
}

par(mfrow=c(1,1))

plot(valeur_seed,Taux_mauvaise_classificaion, type = "l", ylim= c(0,0.6), col = 3, ylab = "Taux")
lines(valeur_seed,Taux_faux_positifs,col=1)
lines(valeur_seed,Taux_faux_negatif,col=2)
legend(90000, 0.6, legend=c("Faux positifs","Faux négatif","Mauvaises classifications"), col=1:3)

```

Variation du taux de mauvaises clasification, de faux positif et de faux négatif



```
# Taux de faux positif
## Maximum
round(max(Taux_faux_positifs), digit =2)
```

```
## [1] 0.08
```

```
## Minimum
round(min(Taux_faux_positifs), digit =2)
```

```
## [1] 0.04
```

```
# Taux de mauvaises classifications
## Maximum
round(max(Taux_mauvaise_classificaion), digit =2 )
```

```
## [1] 0.09
```

```
## Minimum
round(min(Taux_mauvaise_classificaion) , digit = 2)
```

```
## [1] 0.05
```

```
# Taux faux n  gatif
## Maximum
round(max(Taux_faux_negatif), digit = 2)
```

```
## [1] 0.27
```

```
## Minimum
round(min(Taux_faux_negatif), digit = 2)
```

```
## [1] 0.08
```

3.6. for  t al  atoire approche bagging

```
train = train0
test = train0

train$Churn[train$Churn == "False"] = 0
train$Churn[train$Churn == "True"] = 1
train$Churn = as.numeric(train$Churn)

test$Churn[test$Churn == "False"] = 0
test$Churn[test$Churn == "True"] = 1
test$Churn = as.numeric(test$Churn)

library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

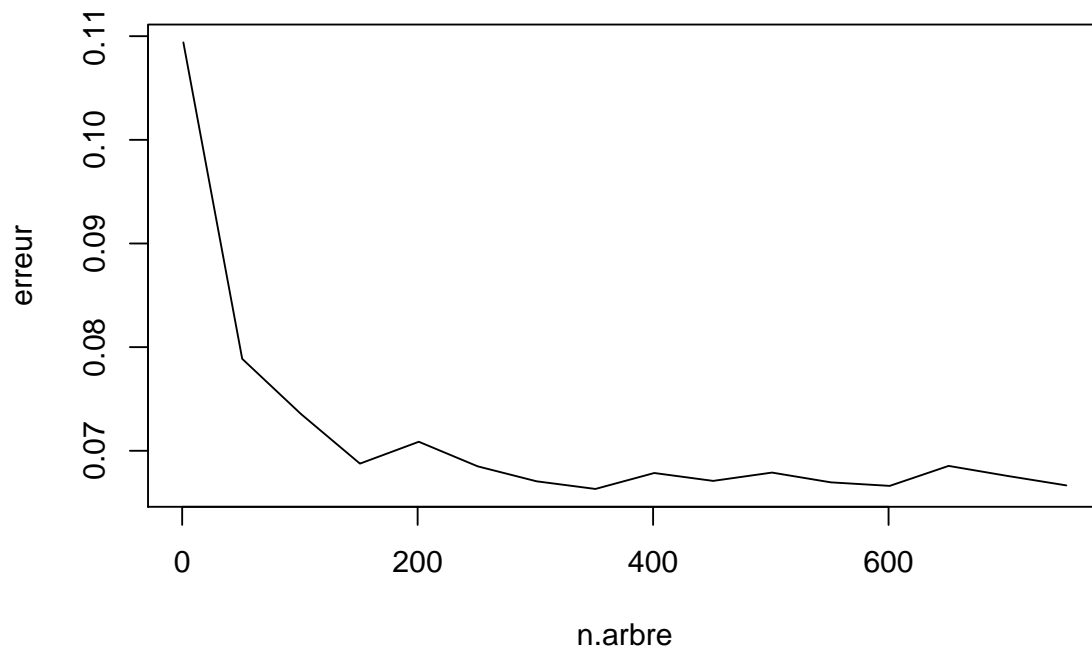
## The following object is masked from 'package:dplyr':
##
##      combine

## The following object is masked from 'package:ggplot2':
##
##      margin

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

mydata$Churn <- as.factor(mydata$Churn)

set.seed(1234)
n.arbre=seq(1,800,by=50)
erreur=NULL
for (i in n.arbre)
{
  rf=randomForest(Churn~.,data=mydata,ntree=i,mtry=length(colnames(mydata))-1)
  erreur=c(erreur,sum(rf$err.rate[,1])/rf$ntree)
}
plot(n.arbre, erreur,type="l")
```



```
rf=randomForest(Churn~.,data=mydata,ntree=600,mtry=length(colnames(mydata))-1)
rf
```

```
##
## Call:
## randomForest(formula = Churn ~ ., data = mydata, ntree = 600,      mtry = length(colnames(mydata))-1)
##              Type of random forest: classification
##              Number of trees: 600
## No. of variables tried at each split: 15
##
##      OOB estimate of  error rate: 6.78%
## Confusion matrix:
##      FALSE TRUE class.error
## FALSE  2759   91  0.03192982
## TRUE    135  348  0.27950311
```

```
importance(rf,type=1)
```

```
##
## State
## Account.length
## Area.code
## International.plan
## Voice.mail.plan
## Number.vmail.messages
## Total.day.minutes
## Total.day.calls
## Total.eve.minutes
```

```
## Total.eve.calls
## Total.night.minutes
## Total.night.calls
## Total.intl.minutes
## Total.intl.calls
## Customer.service.calls
```

3.7. Boosting

```
library(randomForest)
library(adabag)
```

```
## Warning: package 'adabag' was built under R version 4.0.3

## Loading required package: caret

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.0.3

## Loading required package: doParallel

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 4.0.3

## Loading required package: parallel
```

```
train = train0
test = train0

train$Churn[train$Churn == "False"] = 0
train$Churn[train$Churn == "True"] = 1
train$Churn = as.numeric(train$Churn)

test$Churn[test$Churn == "False"] = 0
test$Churn[test$Churn == "True"] = 1
test$Churn = as.numeric(test$Churn)

train$Churn=as.factor(train$Churn)
test$Churn=as.factor(test$Churn)

set.seed(1234)

# boosting with trees of depth 10
myboost=boosting(Churn~., data=train, mfinal = 100, coeflearn = 'Freund', control=rpart.control
myboost$importance
```

```
##      Account.length      Area.code Customer.service.calls
##      4.3438994      0.7133557      5.0969783
##      International.plan Number.vmail.messages      State
##      3.1865928      1.3572300      37.1238850
##      Total.day.calls      Total.day.charge      Total.day.minutes
##      3.6866111      0.0000000      11.8285429
##      Total.eve.calls      Total.eve.charge      Total.eve.minutes
##      3.6300543      0.0000000      7.1423496
##      Total.intl.calls      Total.intl.charge      Total.intl.minutes
##      3.8396138      0.0000000      6.2613481
##      Total.night.calls      Total.night.charge      Total.night.minutes
##      4.4219756      0.0000000      5.5826679
##      Voice.mail.plan
##      1.7848953
```

```
pred=predict(myboost, newdata=test)
```

```
pred$error
```

```
## [1] 0
```

```
M=pred$confusion
```

```
#mytable=table(myboost$class,mydata$Churn)
# names(dimnames(mytable))= c("Predicted", "Observed")
# M = mytable
# M
# a=M[1,1]
# b=M[1,2]
# c=M[2,1]
# d=M[2,2]

# Taux de faux positifs
M[2,1]/(M[2,1]+M[1,1])
```

```
## [1] 0
```

```
# Taux de faux n  gatifs
M[1,2]/(M[2,2]+M[1,2])
```

```
## [1] 0
```

3.8. SVM

4. R  sultats : pr  sentation des r  sultats sous forme de tableaux et figures (ne mettez pas de sortie R)

5. Conclusion/discussion : conclusion g  n  rale, limites de votre   tude, qu'avez vous appris ?