

Detecção de Fraudes no Tráfego de Cliques em Propagandas de Aplicações Mobile

Gilson Santana - gcardososantana@gmail.com

02/11/2020

Informações gerais

Projeto da DSA como parte do treinamento Big Data Analytics com R e Microsoft Azure Machine Learning.

O projeto consiste na criação de modelo de Machine Learning que possa prever se um click para download de aplicativo é ou não fraudulento. Para esse trabalho foi utilizado a base de dados train_sample.csv, disponível no link <https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data>. Maiores detalhe do desafio também pode ser obtido no site do Kaggle. Acesso o arquivo Readme para informações complementares.

Carga & visualização

```
oldw <- getOption("warn")
options(warn = -1)
```

```
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
```

```
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```
library(ggplot2)
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##      combine
#Diretório de trabalho
setwd('D:/OneDrive/EstudosTecnicos/CienciaDados/DtScienceTrab/BigDataRAzure/Cap20/Projeto01')

# Carregar dados - Utilizada a base train-sample.csv devido ao tamanho da base train
dfDados<- read.csv('train_sample.csv', stringsAsFactors= F, header =T)

str(dfDados)

## 'data.frame':    100000 obs. of  8 variables:
## $ ip              : int  87540 105560 101424 94584 68413 93663 17059 121505 192967 143636 ...
## $ app             : int  12 25 12 13 12 3 1 9 2 3 ...
## $ device          : int  1 1 1 1 1 1 1 1 2 1 ...
## $ os              : int  13 17 19 13 1 17 17 25 22 19 ...
## $ channel         : int  497 259 212 477 178 115 135 442 364 135 ...
## $ click_time      : chr   "2017-11-07 09:30:38" "2017-11-07 13:40:27" "2017-11-07 18:05:24" "2017-11-07 18:05:24" ...
## $ attributed_time : chr   "" "" "" "" "" "" "" "" "" "" ...
## $ is_attributed   : int    0 0 0 0 0 0 0 0 0 0 ...

# View(dfDados)
summary(dfDados)

##           ip              app              device              os
## Min.      :    9   Min.    : 1.00   Min.      : 0.00   Min.      : 0.00
## 1st Qu.: 40552   1st Qu.: 3.00   1st Qu.: 1.00   1st Qu.: 13.00
## Median : 79827   Median : 12.00   Median : 1.00   Median : 18.00
## Mean   : 91256   Mean    : 12.05   Mean    : 21.77   Mean    : 22.82
## 3rd Qu.:118252   3rd Qu.: 15.00   3rd Qu.: 1.00   3rd Qu.: 19.00
## Max.    :364757   Max.    :551.00   Max.    :3867.00   Max.    :866.00
## channel    click_time    attributed_time    is_attributed
## Min.      : 3.0   Length:100000   Length:100000   Min.      :0.00000
## 1st Qu.:145.0   Class :character   Class :character   1st Qu.:0.00000
## Median :258.0   Mode  :character   Mode  :character   Median :0.00000
## Mean   :268.8                                     Mean  :0.00227
## 3rd Qu.:379.0                                     3rd Qu.:0.00000
## Max.    :498.0                                     Max.    :1.00000
```

Explorar

Download (e o não download) do App por número de clicks repetidos para diversas variáveis

A quantidade de cliks que não resultam em download para o mesmo IP chama a atenção, sugere fraude. Em outras variáveis essa característica é semelhante, porém não sugestiona fraude na mesma proporção do IP.

```
# ip
#par(mfrow = c(1,2))
dfAux<- dfDados %>%
  select(is_attributed, ip) %>%
  group_by(is_attributed, ip) %>%
  summarise(repeticoes=n())

## `summarise()` regrouping output by 'is_attributed' (override with `groups` argument)

plot01<- dfAux %>% filter(is_attributed == 1) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(ip), repeticoes)) +
```

```

    geom_col() +
    ggtitle('Qtde clicks-IP-APPs baixados-TOP 6') +
    xlab('IPs') +
    ylab('Qtde de clicks por IP')

plot02<- dfAux %>% filter(is_attributed == 0) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(ip), repeticoes)) +
  geom_col() +
  ggtitle('Qtde clicks-IP-APPs NÃO baixados-TOP 6') +
  xlab('IPs') +
  ylab('Qtde de clicks por IP')

# app
dfAux<- dfDados %>%
  select(is_attributed, app) %>%
  group_by(is_attributed, app) %>%
  summarise(repeticoes=n())

## `summarise()` regrouping output by 'is_attributed' (override with `.groups` argument)

plot03<- dfAux %>% filter(is_attributed == 1) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(app), repeticoes)) +
  geom_col() +
  ggtitle('Qtde clicks-Aplicativo-APPs baixados-TOP 6') +
  xlab('Aplicativos') +
  ylab('Qtde de clicks por Aplicativos')

plot04<- dfAux %>% filter(is_attributed == 0) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(app), repeticoes)) +
  geom_col() +
  ggtitle('Qtde clicks-Aplicativo-APPs NÃO baixados-TOP 6') +
  xlab('Aplicativos') +
  ylab('Qtde de clicks por Aplicativos')

# device
dfAux<- dfDados %>%
  select(is_attributed, device) %>%
  group_by(is_attributed, device) %>%
  summarise(repeticoes=n())

## `summarise()` regrouping output by 'is_attributed' (override with `.groups` argument)

plot05<- dfAux %>% filter(is_attributed == 1) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(device), repeticoes)) +
  geom_col() +
  ggtitle('Qtde clicks-Device-APPs baixados-TOP 6') +
  xlab('Device') +
  ylab('Qtde de clicks por Device')

plot06<- dfAux %>% filter(is_attributed == 0) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(device), repeticoes)) +
  geom_col() +
  ggtitle('Qtde de clicks-Device-APPs NÃO baixados-TOP 6') +
  xlab('Device') +

```

```

    ylab('Qtde de clicks por Device')

# OS (Sistema Operacional)
dfAux<- dfDados %>%
  select(is_attributed, os) %>%
  group_by(is_attributed, os) %>%
  summarise(repeticoes=n())

## `summarise()` regrouping output by 'is_attributed' (override with `.groups` argument)

plot07<- dfAux %>% filter(is_attributed == 1) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(os), repeticoes)) +
  geom_col() +
  ggtitle('Qtde clicks-OS-APPS baixados-TOP 6') +
  xlab('Device') +
  ylab('Qtde de clicks por OS')

plot08<- dfAux %>% filter(is_attributed == 0) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(os), repeticoes)) +
  geom_col() +
  ggtitle('Qtde clicks-OS-APPS NÃO baixados-TOP 6') +
  xlab('OS') +
  ylab('Qtde de clicks por OS')

# channel
dfAux<- dfDados %>%
  select(is_attributed, channel) %>%
  group_by(is_attributed, channel) %>%
  summarise(repeticoes=n())

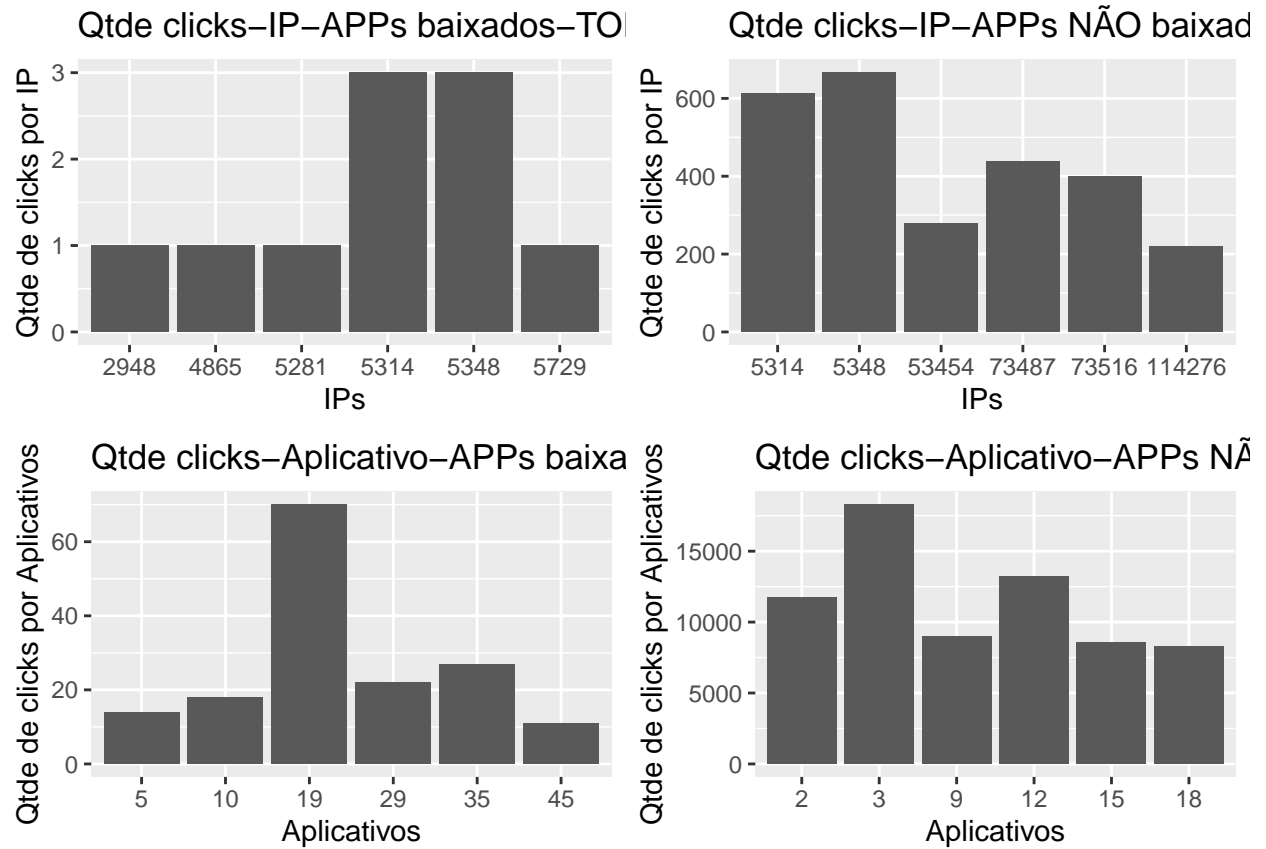
## `summarise()` regrouping output by 'is_attributed' (override with `.groups` argument)

plot09<- dfAux %>% filter(is_attributed == 1) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(channel), repeticoes)) +
  geom_col() +
  ggtitle('Qtde clicks-Channel-APPS baixados-TOP 6') +
  xlab('Channel') +
  ylab('Qtde de clicks por Channel')

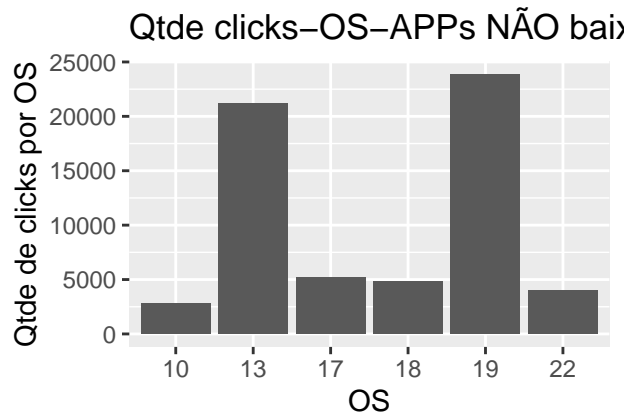
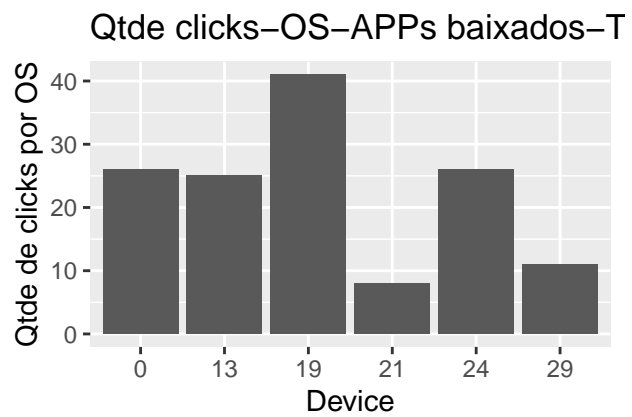
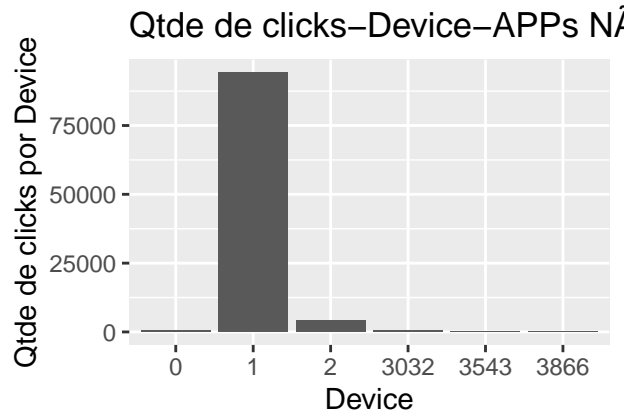
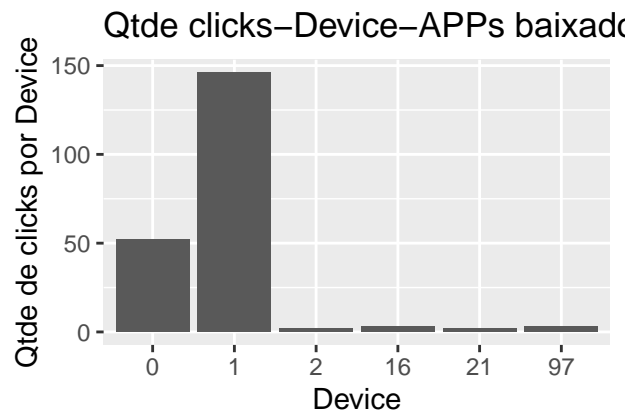
plot10<- dfAux %>% filter(is_attributed == 0) %>% arrange(desc(repeticoes)) %>% head(6) %>%
  ggplot(aes(factor(channel), repeticoes)) +
  geom_col() +
  ggtitle('Qtde clicks-Channel-APPS NÃO baixados-TOP 6') +
  xlab('Channel') +
  ylab('Qtde de clicks por Channel')

#Exibir plots
grid.arrange(plot01, plot02, plot03, plot04, ncol=2, nrow=2)

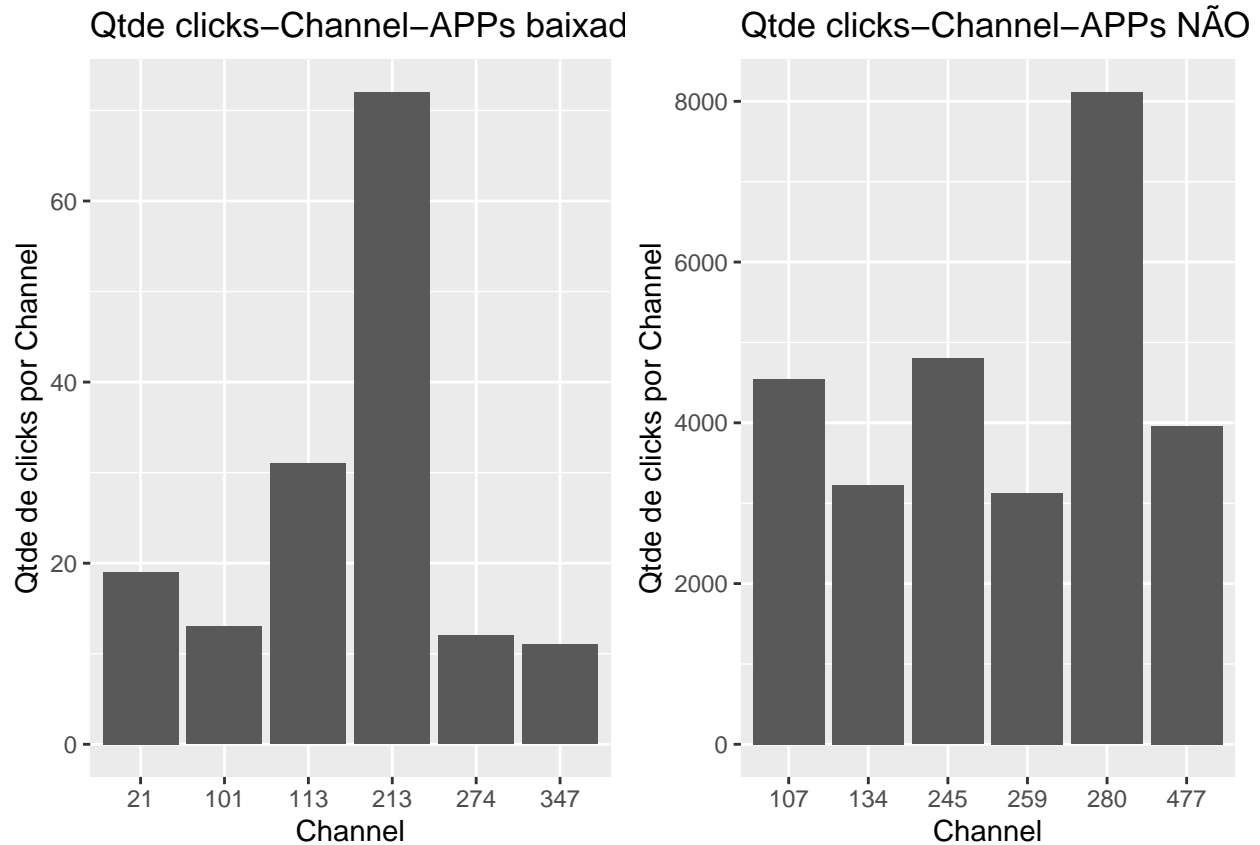
```



```
grid.arrange(plot05, plot06, plot07, plot08, ncol=2, nrow=2)
```



```
grid.arrange(plot09, plot10, ncol=2, nrow=1)
```



Relação attributed_time X is_attributed

Se App baixado implica na existência do attributed_time, assim essa variável é consequência da variável target (is_attributed), não pode figurar como perditora

```
nrow(dfDados %>% filter(attributed_time == ''))
```

```
## [1] 99773
```

```
nrow(dfDados %>% filter(is_attributed == '0'))
```

```
## [1] 99773
```

```
nrow(dfDados %>% filter(attributed_time == '' & is_attributed == '0'))
```

```
## [1] 99773
```

```
dfDados$attributed_time <- NULL
```

Análise temporal

Variável click_time - indica o momento do click.

```
dfDados$dt <- date(dfDados$click_time)
dfDados$year <- year(dfDados$dt)
dfDados$month <- month(dfDados$dt)
dfDados$day <- day(dfDados$dt)
dfDados$weekday <- wday(dfDados$dt)
```

```

dfDados$hour <- hour(dfDados$click_time)
str(dfDados)

## 'data.frame': 100000 obs. of 13 variables:
## $ ip : int 87540 105560 101424 94584 68413 93663 17059 121505 192967 143636 ...
## $ app : int 12 25 12 13 12 3 1 9 2 3 ...
## $ device : int 1 1 1 1 1 1 1 1 2 1 ...
## $ os : int 13 17 19 13 1 17 17 25 22 19 ...
## $ channel : int 497 259 212 477 178 115 135 442 364 135 ...
## $ click_time : chr "2017-11-07 09:30:38" "2017-11-07 13:40:27" "2017-11-07 18:05:24" "2017-11-07
## $ is_attributed: int 0 0 0 0 0 0 0 0 0 0 ...
## $ dt : Date, format: "2017-11-07" "2017-11-07" ...
## $ year : num 2017 2017 2017 2017 2017 ...
## $ month : num 11 11 11 11 11 11 11 11 11 11 ...
## $ day : int 7 7 7 7 9 9 9 7 8 8 ...
## $ weekday : num 3 3 3 3 5 5 5 3 4 4 ...
## $ hour : int 9 13 18 4 9 1 1 10 9 12 ...

# year e month - essas variáveis tem uma dimensão apenas, é
# como uma constante. A amostra se resume a um ano e um mês, apenas
str(factor(dfDados$year))

## Factor w/ 1 level "2017": 1 1 1 1 1 1 1 1 1 1 ...

str(factor(dfDados$month))

## Factor w/ 1 level "11": 1 1 1 1 1 1 1 1 1 1 ...

dfDados$year <- NULL
dfDados$month <- NULL

# day e weekday - se tornaram redundantes devido ao tamanho da amostra: apenas
# 4 dias consecutivos no mês 11 de 2017, sendo que não contempla fim de semana
# nem sinaliza feriados. Qualquer análise de comportamento na linha semanal ou
# de períodos mensais seria "forçação" de barra.
# Foi mantida a variável day e eliminada a weekday
str(factor(dfDados$day))

## Factor w/ 4 levels "6","7","8","9": 2 2 2 2 4 4 4 2 3 3 ...

dfDados %>% distinct(day) %>% arrange(day)

## day
## 1 6
## 2 7
## 3 8
## 4 9

str(factor(dfDados$weekday))

## Factor w/ 4 levels "2","3","4","5": 2 2 2 2 4 4 4 2 3 3 ...

dfDados %>% distinct(weekday) %>% arrange(weekday)

## weekday
## 1 2
## 2 3
## 3 4

```


4

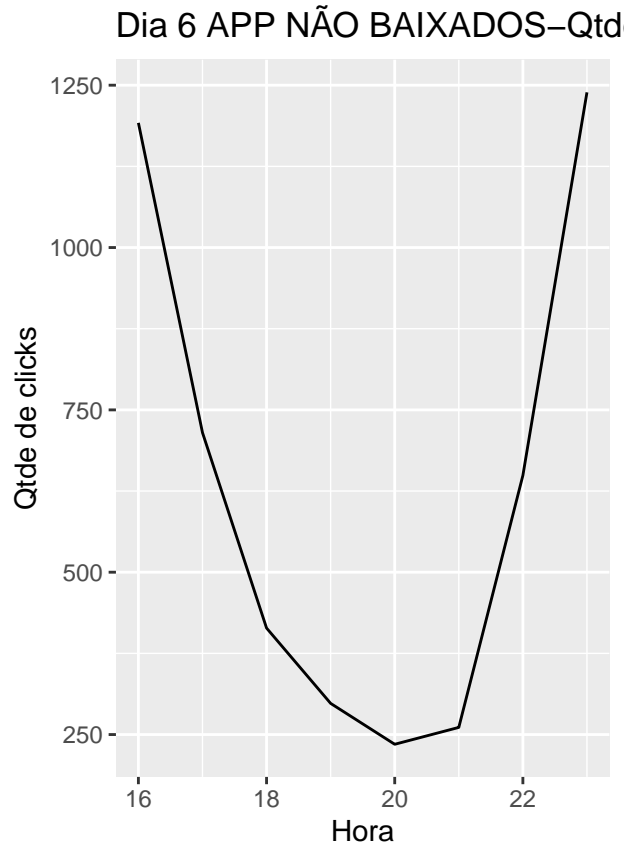
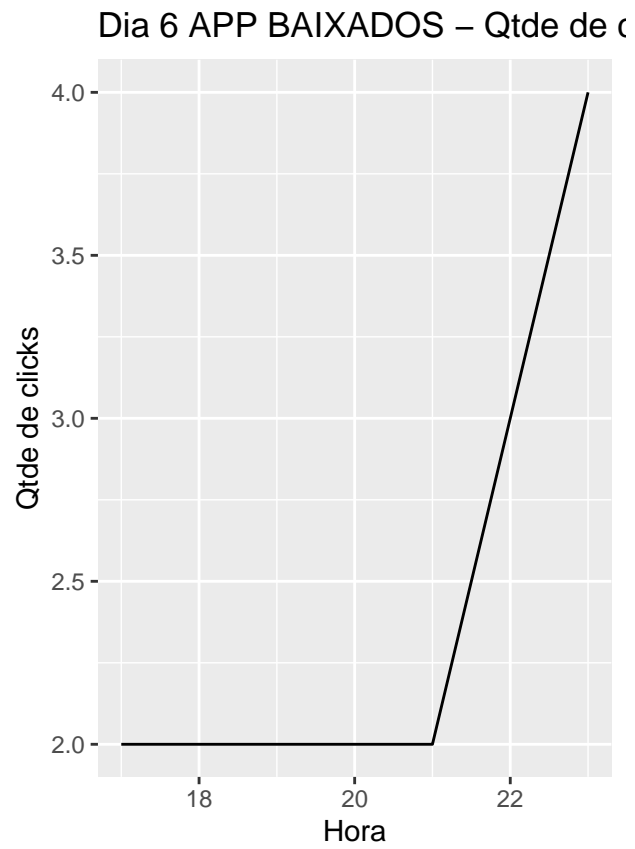
5

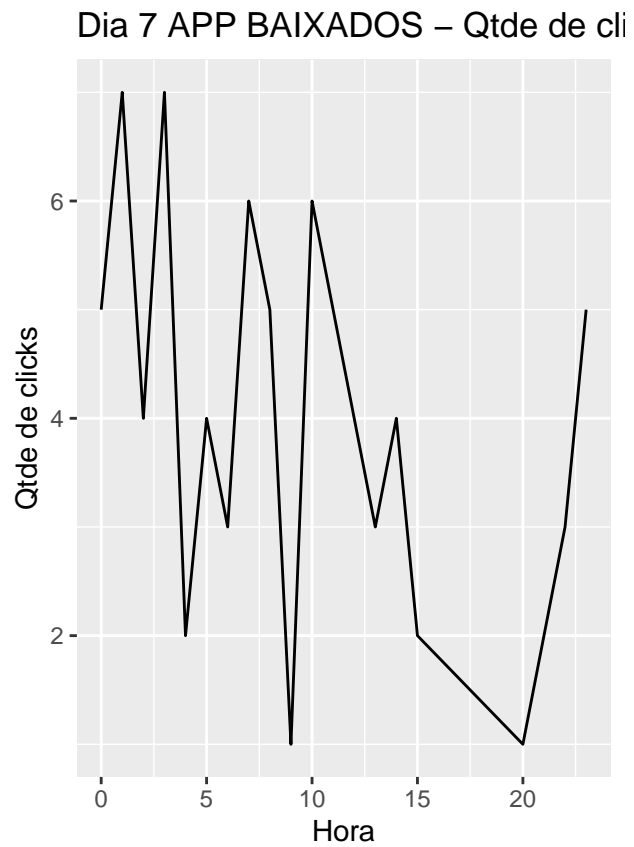
```
dfDados$weekday <- NULL
```

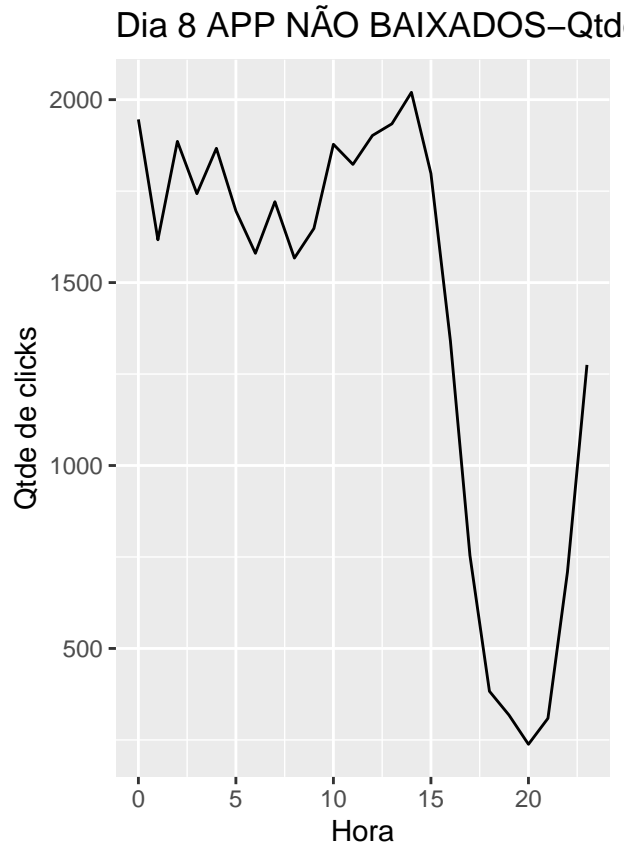
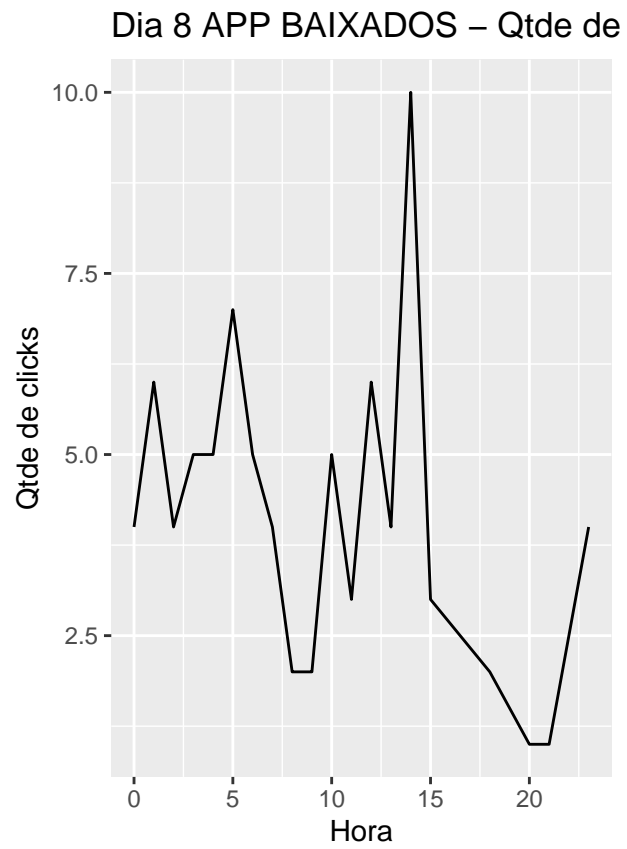
```
# hour - Avaliar comportamento por hora  
# O desbalanceamento da classe is_attributed é muito acentuado, isso limita os insights  
# Recomenda-se repetir a análise após o balanceamento da classe. Poderá ser feito em uma  
# continuidade desse trabalho.  
# Para um análise temporal, a amostra se apresentou bastante limitada.  
dfAux<- dfDados %>%  
  select(is_attributed, day, hour) %>%  
  group_by(is_attributed, day, hour) %>%  
  summarise(qtdCliks=n())
```

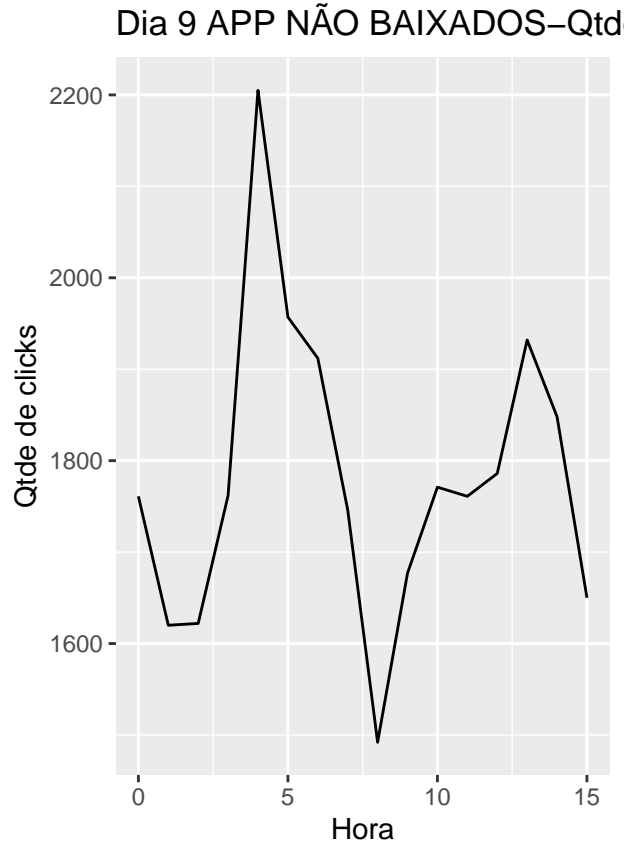
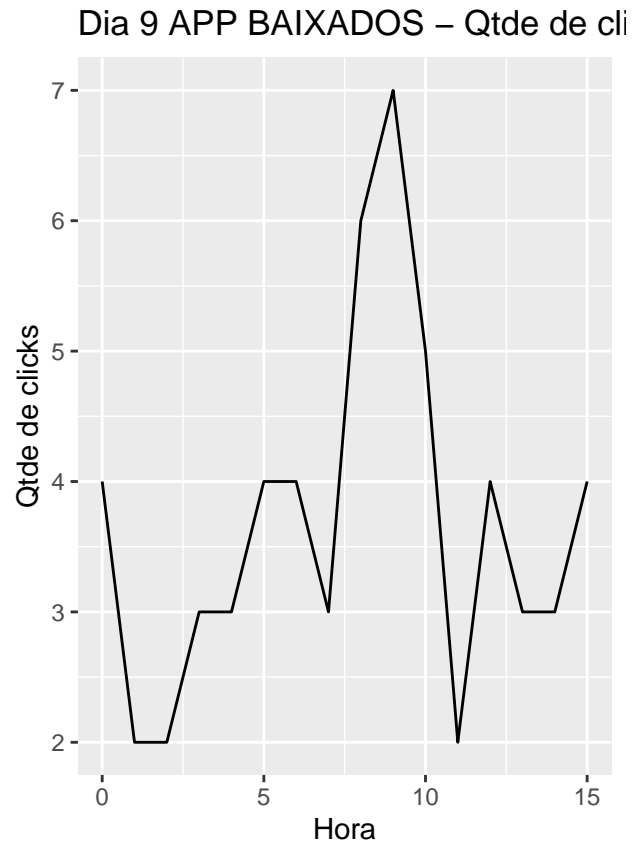
`summarise()` regrouping output by 'is_attributed', 'day' (override with `.groups` argument)

```
# Função - plotagem por dia  
plotD <- function(dia){  
  plotd1<- ggplot(dfAux[dfAux$is_attributed == 1 & dfAux$day == dia,], aes(x = hour, y = qtdCliks)) +  
    geom_line() +  
    ylab("Qtde de clicks") + xlab("Hora") +  
    labs(title = paste("Dia ", as.character(dia), " APP BAIXADOS - Qtde de clicks por hora",  
  plotd2<- ggplot(dfAux[dfAux$is_attributed == 0 & dfAux$day == dia,], aes(x = hour, y = qtdCliks)) +  
    geom_line() +  
    ylab("Qtde de clicks") + xlab("Hora") +  
    labs(title = paste("Dia ", as.character(dia), " APP NÃO BAIXADOS-Qtde de clicks por hora",  
  grid.arrange(plotd1 , plotd2, ncol=2, nrow=1)  
}  
  
dias<- unlist(distinct(dfAux[, 'day']))  
lapply(dias, plotD)
```









```
## $day1
## TableGrob (1 x 2) "arrange": 2 grobs
##   z      cells      name      grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
##
## $day2
## TableGrob (1 x 2) "arrange": 2 grobs
##   z      cells      name      grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
##
## $day3
## TableGrob (1 x 2) "arrange": 2 grobs
##   z      cells      name      grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
##
## $day4
## TableGrob (1 x 2) "arrange": 2 grobs
##   z      cells      name      grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

Análise de correlação

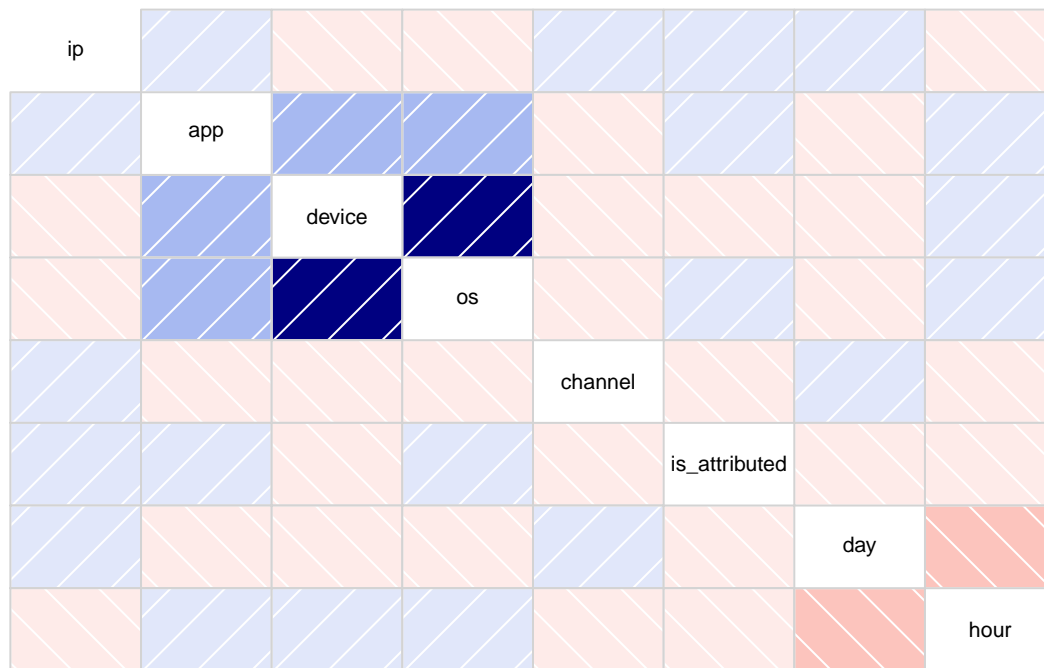
```
library(corrgram)
```

```
## Registered S3 method overwritten by 'seriation':
```

```
##   method      from
```

```
##   reorder.hclust gclus
```

```
corrgram(dfDados)
```



Variáveis categóricas

Reconhecer as variáveis categóricas do dataset. Foram disponibilizadas com o tipo inteiro

```
toFactor<- function(df, var) {  
  for(v in var) df[,v]= factor(df[,v])  
  return(df)  
}
```

```
VarToFactor<- c('ip', 'app','device','os','channel','is_attributed', 'day', 'hour')  
dfDados<- toFactor(dfDados, VarToFactor)
```

Valores missing

Checar e tratar valores missing

```
# Tratar valores NA - Não tem valores missing  
any(is.na(dfDados))
```

```
## [1] FALSE
```

Dados de teste e treino

```
#Divisão dos dados de teste e treino
linhas <- sample(1:nrow(dfDados), 0.7 * nrow(dfDados))
dfTrain <- dfDados[linhas,]
dfTest  <- dfDados[-linhas,]
```

Balanceamento de classe

O dataset está bastante desbalanceado em relação a classe is_attributed, variável target.

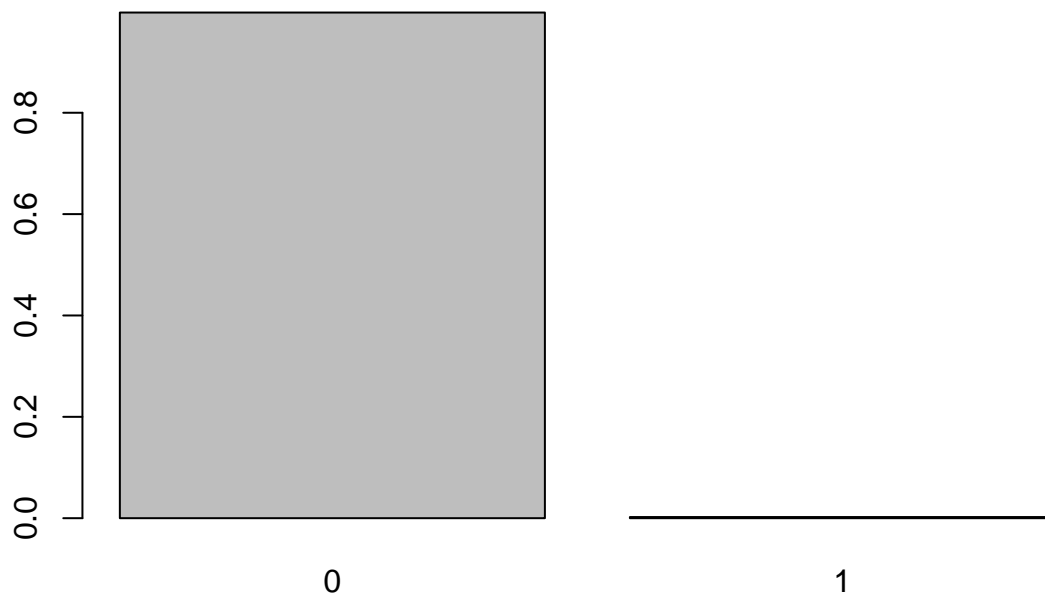
```
#Balanceamento - Oversampling - is_attributed - muito mais registros com valor 0
summary(dfTrain$is_attributed)
```

```
##      0      1
## 69843  157
```

```
prop.table(table(dfTrain$is_attributed))
```

```
##
##           0           1
## 0.997757143 0.002242857
```

```
barplot(prop.table(table(dfTrain$is_attributed)))
```



```
dfTrainBal <- ROSE(is_attributed ~ ip +
                  app +
                  device +
                  os +
                  channel +
                  day +
                  hour,
                  data = dfTrain, seed = 1)$data

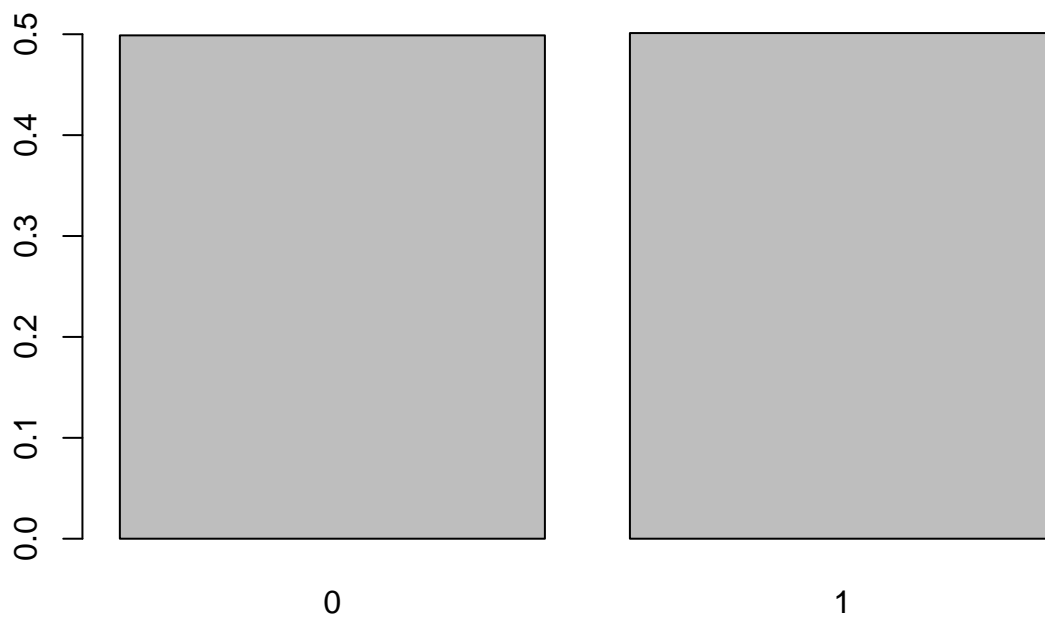
# Nova Proporção
summary(dfTrainBal$is_attributed)
```

```
##      0      1
## 34919 35081
```

```
prop.table(table(dfTrainBal$is_attributed))
```

```
##
##      0      1
## 0.4988429 0.5011571
```

```
barplot(prop.table(table(dfTrainBal$is_attributed)))
```



```
any(is.na(dfTrainBal))
```

```
## [1] FALSE
```



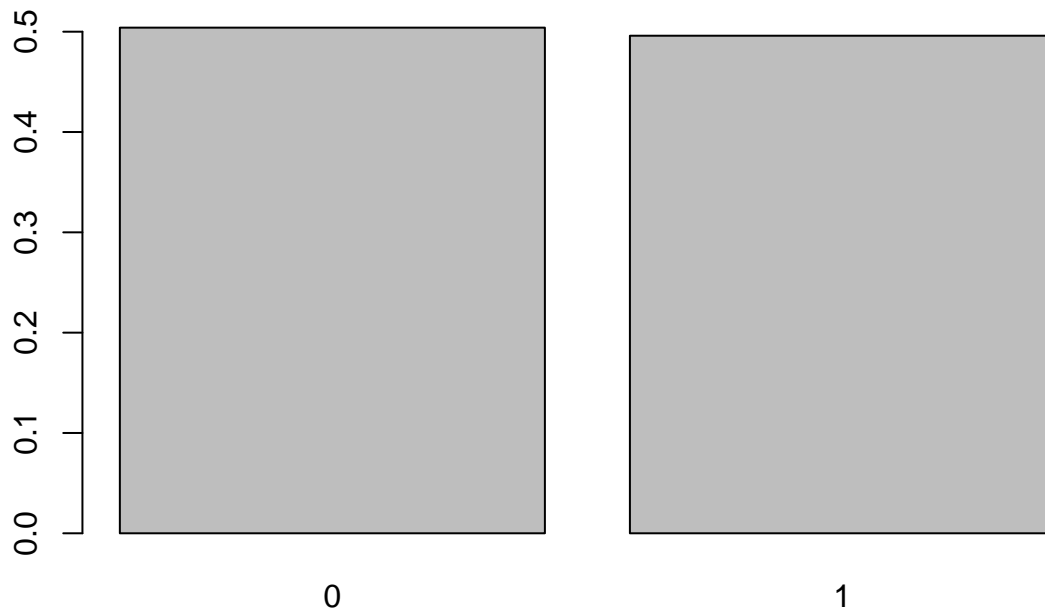
```
# Balancear Teste
dfTestBal <- ROSE(is_attributed ~ ip +
                  app +
                  device +
                  os +
                  channel +
                  day +
                  hour,
                  data = dfTest, seed = 1)$data
summary(dfTestBal$is_attributed)
```

```
##      0      1
## 15121 14879
```

```
prop.table(table(dfTestBal$is_attributed))
```

```
##
##           0           1
## 0.5040333 0.4959667
```

```
barplot(prop.table(table(dfTestBal$is_attributed)))
```



```
any(is.na(dfTestBal))
```

```
## [1] FALSE
```

Treinando modelos

OBS: O Radom Forest não aceita trabalhar com variáveis categóricas com mais de 53 níveis. Por esse motivo, algumas Variáveis foram ajustadas para o tipo character.

```
library(C50) #algoritmo C5.0
library(e1071) #naiveBayes
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:gridExtra':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
##
## The following object is masked from 'package:dplyr':
##
##     combine

#Árvore de decisão - algoritmo C5.0
m.Arvore1 <- C5.0(is_attributed ~ ., data = dfTrainBal, rules = TRUE)

# naiveBayes
m.Naive1<- naiveBayes(is_attributed ~ ., data = dfTrainBal, laplace = 0)

# Radom Forest - Não aceita trabalhar com factor com mais de 53 níveis. Variáveis
# ajustadas para o tipo character.
dfTrainBalRandom<- dfTrainBal
dfTrainBalRandom$ip<- as.character(dfTrainBalRandom$ip)
dfTrainBalRandom$app<- as.character(dfTrainBalRandom$app)
dfTrainBalRandom$device<- as.character(dfTrainBalRandom$device)
dfTrainBalRandom$os<- as.character(dfTrainBalRandom$os)
dfTrainBalRandom$channel<- as.character(dfTrainBalRandom$channel)
#str(dfTrainBalRandom)

m.Random1 <- randomForest( is_attributed ~ ip +
                           app +
                           device +
                           os +
                           channel +
                           day +
                           hour,
                           data = dfTrainBalRandom,
                           ntree = 100, nodesize = 10)
```

Predição e avaliação

Predições

```
p.Arvore1<- predict(m.Arvore1, dfTestBal)
p.Naive1<- predict(m.Naive1, dfTestBal)

dfTestBalRandom<- dfTestBal
dfTestBalRandom$ip<- as.character(dfTestBalRandom$ip)
dfTestBalRandom$app<- as.character(dfTestBalRandom$app)
dfTestBalRandom$device<- as.character(dfTestBalRandom$device)
dfTestBalRandom$os<- as.character(dfTestBalRandom$os)
dfTestBalRandom$channel<- as.character(dfTestBalRandom$channel)
p.Random1<- predict(m.Random1, dfTestBalRandom)
```

#Avaliando predições

```
library(caret)
```

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

```
##
```

```
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:corrgram':
```

```
##
```

```
## panel.fill
```

```
confusionMatrix(dfTestBal$is_attributed, p.Arvore1)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction    0    1
```

```
##           0 13942 1179
```

```
##           1  3821 11058
```

```
##
```

```
##           Accuracy : 0.8333
```

```
##           95% CI : (0.8291, 0.8375)
```

```
## No Information Rate : 0.5921
```

```
## P-Value [Acc > NIR] : < 2.2e-16
```

```
##
```

```
##           Kappa : 0.6662
```

```
##
```

```
## McNemar's Test P-Value : < 2.2e-16
```

```
##
```

```
##           Sensitivity : 0.7849
```

```
##           Specificity : 0.9037
```

```
## Pos Pred Value : 0.9220
```

```
## Neg Pred Value : 0.7432
```

```
## Prevalence : 0.5921
```

```
## Detection Rate : 0.4647
```

```
## Detection Prevalence : 0.5040
```

```
## Balanced Accuracy : 0.8443
```

```
##
```

```
## 'Positive' Class : 0
```

```
##
```

```
confusionMatrix(dfTestBal$is_attributed, p.Naive1)
```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 14185   936
##           1  1742 13137
##
##           Accuracy : 0.9107
##           95% CI : (0.9075, 0.9139)
##       No Information Rate : 0.5309
##       P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.8214
##
##  McNemar's Test P-Value : < 2.2e-16
##
##       Sensitivity : 0.8906
##       Specificity : 0.9335
##       Pos Pred Value : 0.9381
##       Neg Pred Value : 0.8829
##       Prevalence : 0.5309
##       Detection Rate : 0.4728
##       Detection Prevalence : 0.5040
##       Balanced Accuracy : 0.9121
##
##       'Positive' Class : 0
##
confusionMatrix(dfTestBal$is_attributed, p.Random1)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 15060   61
##           1 14879    0
##
##           Accuracy : 0.502
##           95% CI : (0.4963, 0.5077)
##       No Information Rate : 0.998
##       P-Value [Acc > NIR] : 1
##
##           Kappa : -0.0041
##
##  McNemar's Test P-Value : <2e-16
##
##       Sensitivity : 0.5030
##       Specificity : 0.0000
##       Pos Pred Value : 0.9960
##       Neg Pred Value : 0.0000
##       Prevalence : 0.9980
##       Detection Rate : 0.5020
##       Detection Prevalence : 0.5040
##       Balanced Accuracy : 0.2515
##

```

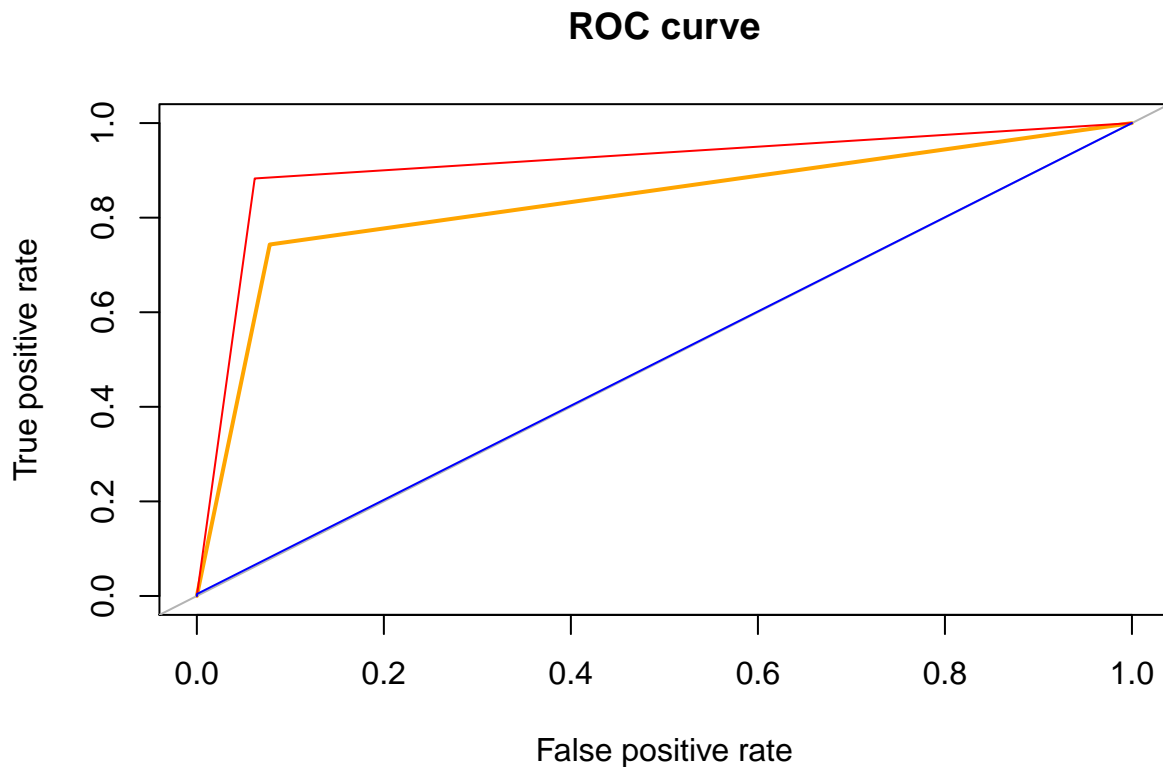
```
##          'Positive' Class : 0
##
# ROC Curves com o ROSE
roc.curve(dfTestBal$sis_attributed, p.Arvore1, plotit = T, col = "orange", add.roc = F)

## Area under the curve (AUC): 0.833

roc.curve(dfTestBal$sis_attributed, p.Naive1, plotit = T, col = "red", add.roc = T)

## Area under the curve (AUC): 0.911

roc.curve(dfTestBal$sis_attributed, p.Random1, plotit = T, col = "blue", add.roc = T)
```



```
## Area under the curve (AUC): 0.502
options(warn = oldw)
```

Conclusões

O modelo baseado no Naive apresentou melhor acurácia, poderia seguir com um refinamento do processo de otimização.

O C5.0 vem em seguida. Apresenta também como candidato a seguir com uma otimização.

O modelo de Radom Forest não conseguiu rodar com as variáveis categóricas (factor) com mais de 53 níveis. Assim, algumas variáveis foram convertidas para caracter. isso pode ter influenciando no seu baixo desempenho.