AdTracking Fraude Detection

Fernando Tsutomu Hara

6/21/2020

AdTracking Fraude Detection

Este projeto tem como objetivo construir um algoritmo que prevê se um usuário fará o download de um aplicativo após clicar na propaganda do app. O arquivo de suporte de treinos tem aproximadamente 7 Gb, o que torna praticamente inviável de se trabalhar no R, mas também possui uma amostra de aproximadamente 4 Mb que será usado como base nessa predição. Como o arqivo é pequeno comparado ao original, não conseguimos extrair toda amostra necessária, mas já da para prever com uma acurácia boa.

Neste projeto foram feitos 4 previsões para o kaggle. Então temos basicamente entre todas as previsões uma etapa de organização dos dados, modelagem preditiva e por último a modificação no arquivo de teste.

Leitura do Arquivo e Análise e Limpeza dos dados

Para a primeira previsão, que será considerada nossa baseline, vamos fazer uma análise e limpeza mais robusta na nossa amostra, a fim de entender melhor como os dados estão distribuídos e se comportam em relação à variável target.

Primeiro iremos ler a amostra e ver um resumo geral das variáveis, depois inicializaremos a etapa de análise e limpeza dos dados, realizando uma feature selection, daí partiremos para a primeira modelagem preditiva.

```
library(data.table)
# lendo o arquivo de treino
df <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_com_R/master/
# Visualização do Data Frame
head(df)
```

```
##        ip app device os channel          click_time attributed_time
## 1:  87540  12      1 13     497 2017-11-07 09:30:38
## 2: 105560  25      1 17     259 2017-11-07 13:40:27
## 3: 101424  12      1 19     212 2017-11-07 18:05:24
## 4:  94584  13      1 13     477 2017-11-07 04:58:08
## 5:  68413  12      1  1     178 2017-11-09 09:00:09
## 6:  93663   3      1 17     115 2017-11-09 01:22:13
##    is_attributed
## 1:             0
## 2:             0
## 3:             0
## 4:             0
## 5:             0
## 6:             0
```

```
# Resumo do Data Frame
str(df)
```

```
## Classes 'data.table' and '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" "2017-11-0
##  $ attributed_time: chr  "" "" "" "" ...
##  $ is_attributed  : int  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Criando uma função para transformar variáveis em fatores
convert_factor <- function(df, variaveis){
  for(variavel in variaveis){
    df[[variavel]] = as.factor(df[[variavel]])
  }
  return(df)
}

# A variável ip é apenas um nome do, portanto não será utilizada em meu modelo preditivo, mas será tran
# A variável target "is_attributed" também será tranformada para fator.
# Criando o vetor de variaveis
variaveis <- c('ip', 'is_attributed')
# Chamando a função
df <- convert_factor(df, variaveis)

# Verificando se o df tem algum valor na.
sum(is.na(df) == TRUE)
```

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

```
# Vamos separar o "click_time" entre duas variáveis data e hora, mantendo a variável e depois alterar o
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

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

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

```
library(tidyr)
df$click_time2 <- df$click_time
df <- df %>%
  separate(click_time2, c("click_date", "click_hour"), " ")

# Convertendo click_date para o tipo Data
df$click_date <- as.Date(df$click_date)

# Convertendo click_hour para o tipo Hora
#install.packages("hms")
```

```
library(hms)
df$click_hour <- as_hms(df$click_hour)

# Convertendo click_time para o tipo POSIXct
df$click_time <- as.POSIXct(df$click_time)

# Conferindo novamente os dados.
str(df)
```

```
## Classes 'data.table' and 'data.frame':   100000 obs. of  10 variables:
##  $ ip             : Factor w/ 34857 levels "9","10","19",..: 15221 18449 17664 16497 11853 16301 297
##  $ 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     : POSIXct, format: "2017-11-07 09:30:38" "2017-11-07 13:40:27" ...
##  $ attributed_time: chr  "" "" "" "" ...
##  $ is_attributed  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ click_date     : Date, format: "2017-11-07" "2017-11-07" ...
##  $ click_hour     : 'hms' num  09:30:38 13:40:27 18:05:24 04:58:08 ...
##   ..- attr(*, "units")= chr "secs"
##  - attr(*, ".internal.selfref")=<externalptr>
```

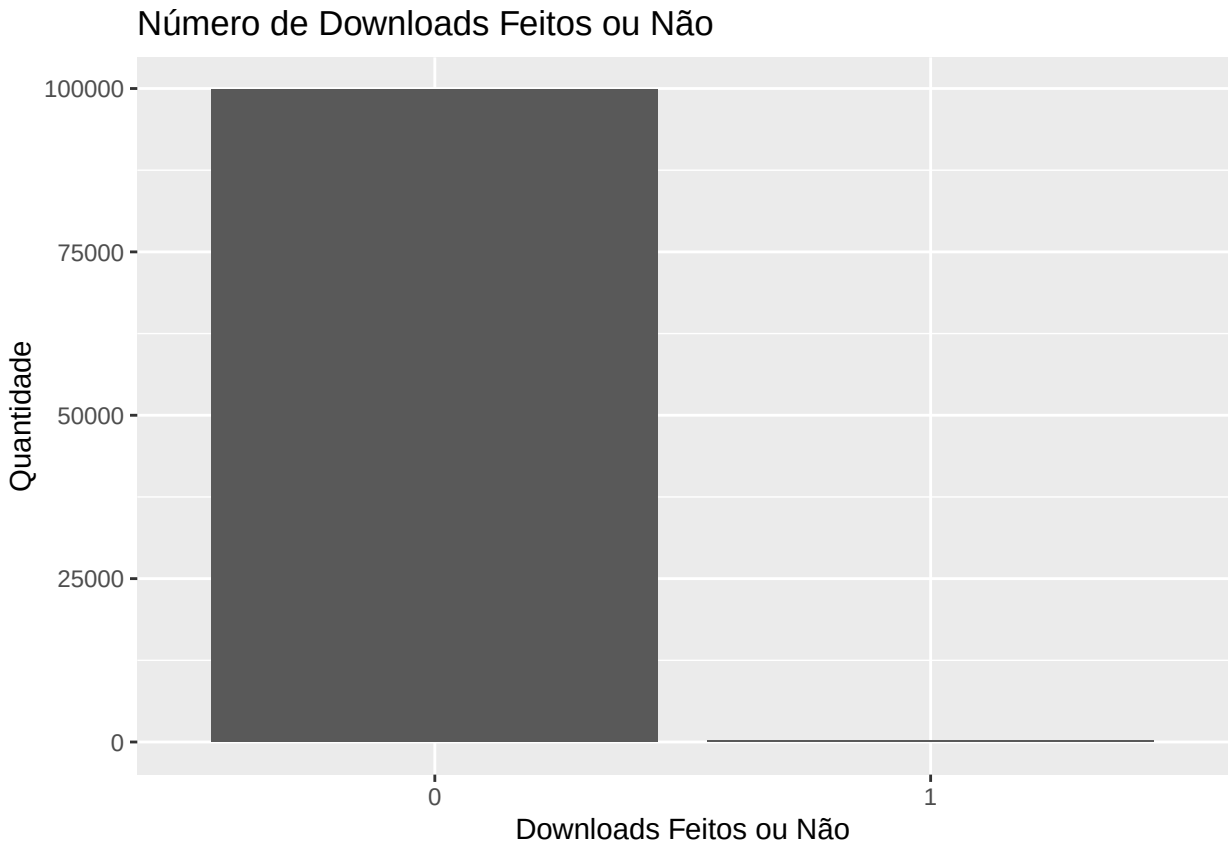
```
# Agora podemos fazer a exploração dos dados.
# Analizando a variável target
table(df$is_attributed)
```

```
##
##     0     1
## 99773   227
```

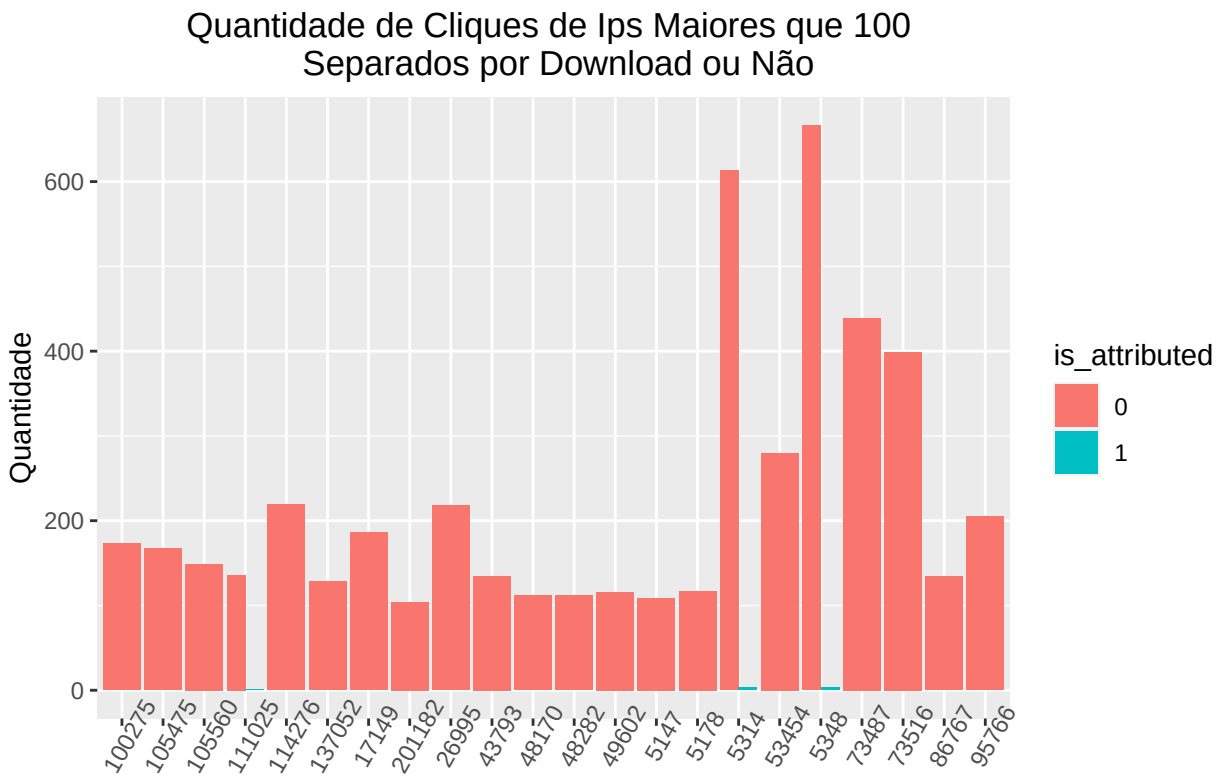
```
prop.table(table(df$is_attributed))
```

```
##
##       0       1
## 0.99773 0.00227
```

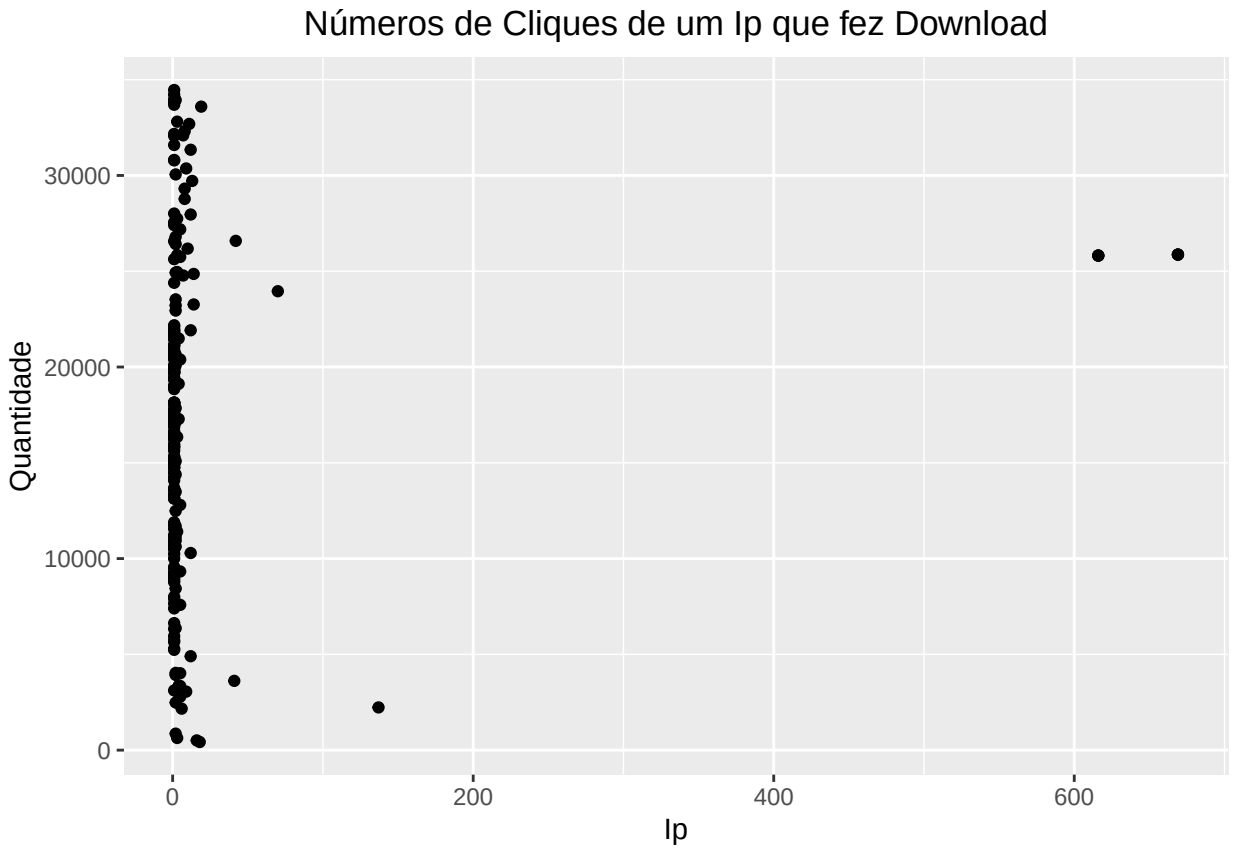
```
library(ggplot2)

ggplot(df, aes(x=is_attributed, ..count..)) +
  geom_bar() +
  ggtitle("Número de Downloads Feitos ou Não") +
  xlab("Downloads Feitos ou Não") +
  ylab("Quantidade")
```




```
# A diferença entre o número de downloads feitos e não feitos é muito grande, vamos utilizar o método R

# Como a variável attributed_time não existe nos outros data sets e também é uma "resposta" do algoritm
df$attributed_time <- NULL

# Explorando a variável ip
# A variável ip é apenas o nome da maquina, por isso, dificilmente conseguimos extrair algo disso na mo
df$ip <- as.character(df$ip)
# Criando a variável count_ip
df$count_ip <- as.numeric(ave(df$ip, df$ip, FUN = length))
df$ip <- as.factor(df$ip)

# Agora podemos fazer um gráfico para ver como os ips com números grandes de cliques se comportam.
df %>%
  filter(count_ip > 100) %>%
  ggplot(aes(ip, ..count..)) +
  geom_bar(aes(fill = is_attributed), position = "dodge") +
  ggtitle("Quantidade de Cliques de Ips Maiores que 100 \n Separados por Download ou Não") +
  xlab("Ips com mais de 100 cliques") +
  ylab("Quantidade") +
  theme(axis.text.x = element_text(angle = 60),
        plot.title = element_text(hjust = 0.5))
```




```
# Podemos ver que existem pouquíssimos downloads realizados por quem realiza muitos cliques.

# Filtrando por downloads feitos e fazendo o gráfico para ver o número de cliques de um ip que fez down
df %>%
  filter(is_attributed == 1) %>%
  ggplot(aes(x=count_ip, y=as.numeric(ip))) +
  geom_point() +
  ggtitle("Números de Cliques de um Ip que fez Download") +
  xlab("Ip") +
  ylab("Quantidade") +
  theme(plot.title = element_text(hjust = 0.5))
```



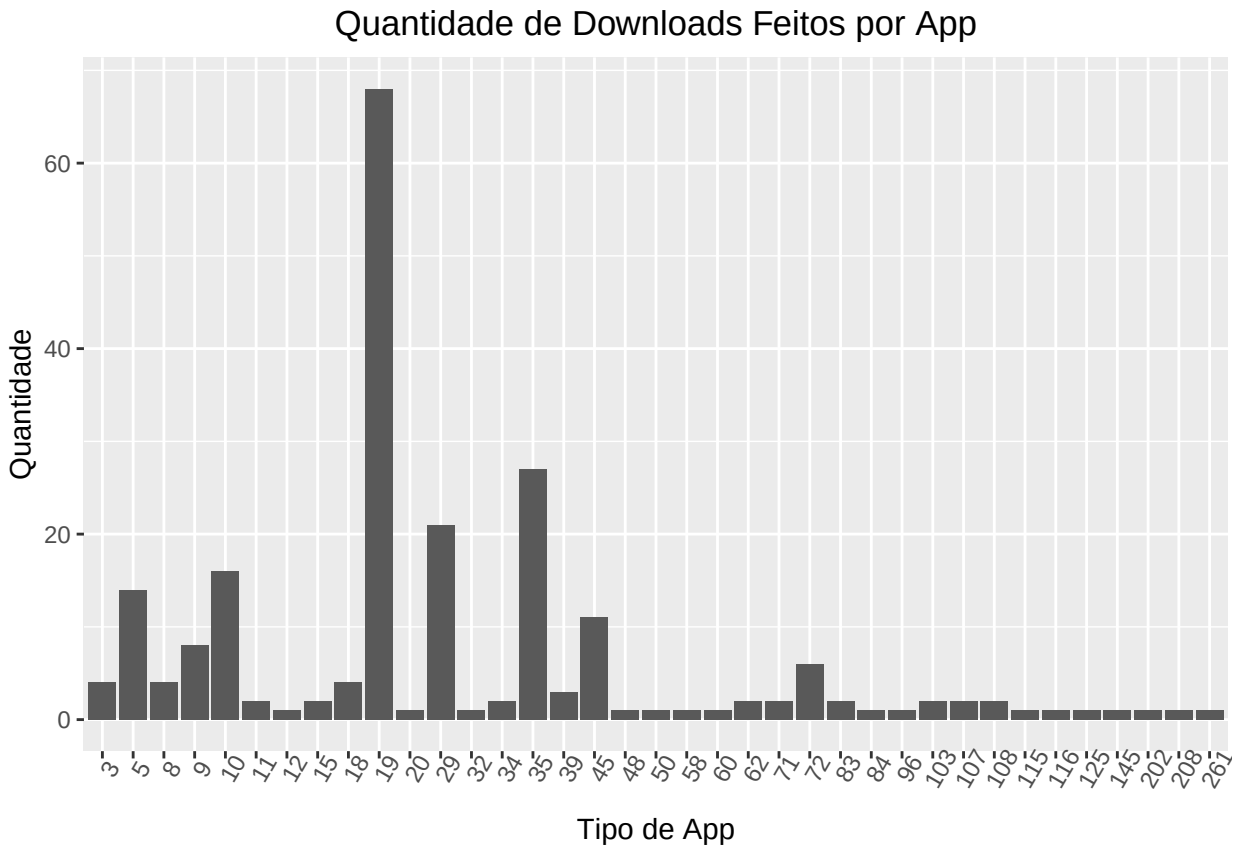

```
# Vamos filtrar para ver quantos cliques acima de 100 que fizeram o download.
nrow(df %>%
       filter(is_attributed == 1) %>%
       filter(count_ip > 100))
```

```
## [1] 7
```

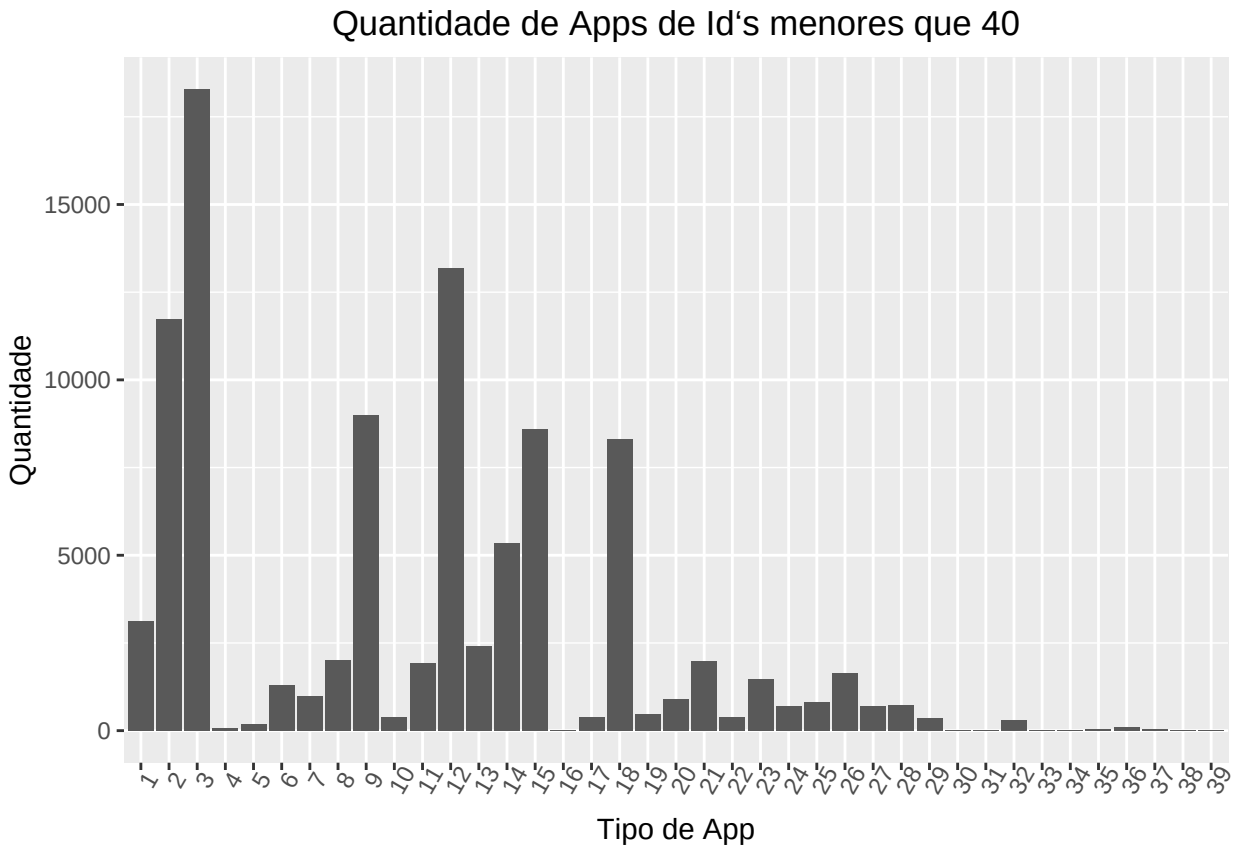
```
# Temos apenas 7 downloads feito quando o número de cliques é maior que 100. Esses números são consider
df <- df[!(df$is_attributed==1 & df$count_ip>100)]

nrow(df[(df$is_attributed == 1 & df$count_ip == 1)])
```

```
## [1] 150
```

```
# Em números podemos ver que quase mais da metade de downloads feitos foram por quem clicou apenas 1 ve

# Analizando a variável app.
df %>%
  filter(is_attributed == 1) %>%
  ggplot(aes(x=as.factor(app), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Downloads Feitos por App") +
    xlab("Tipo de App") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```




```
# Como podemos ver, a maioria dos downloads foram feitos pelo app do tipo 19 e também tem outros apps q
df %>%
  filter(app < 40) %>%
  ggplot(aes(as.factor(app), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Apps de Id`s menores que 40") +
    xlab("Tipo de App") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```




```
# Analizando esses dois gráficos podemos ver que os apps do tipo 19 e 35 não são muitos considerados a
```

```
nrow(df[df$app==19])
```

```
## [1] 476
```

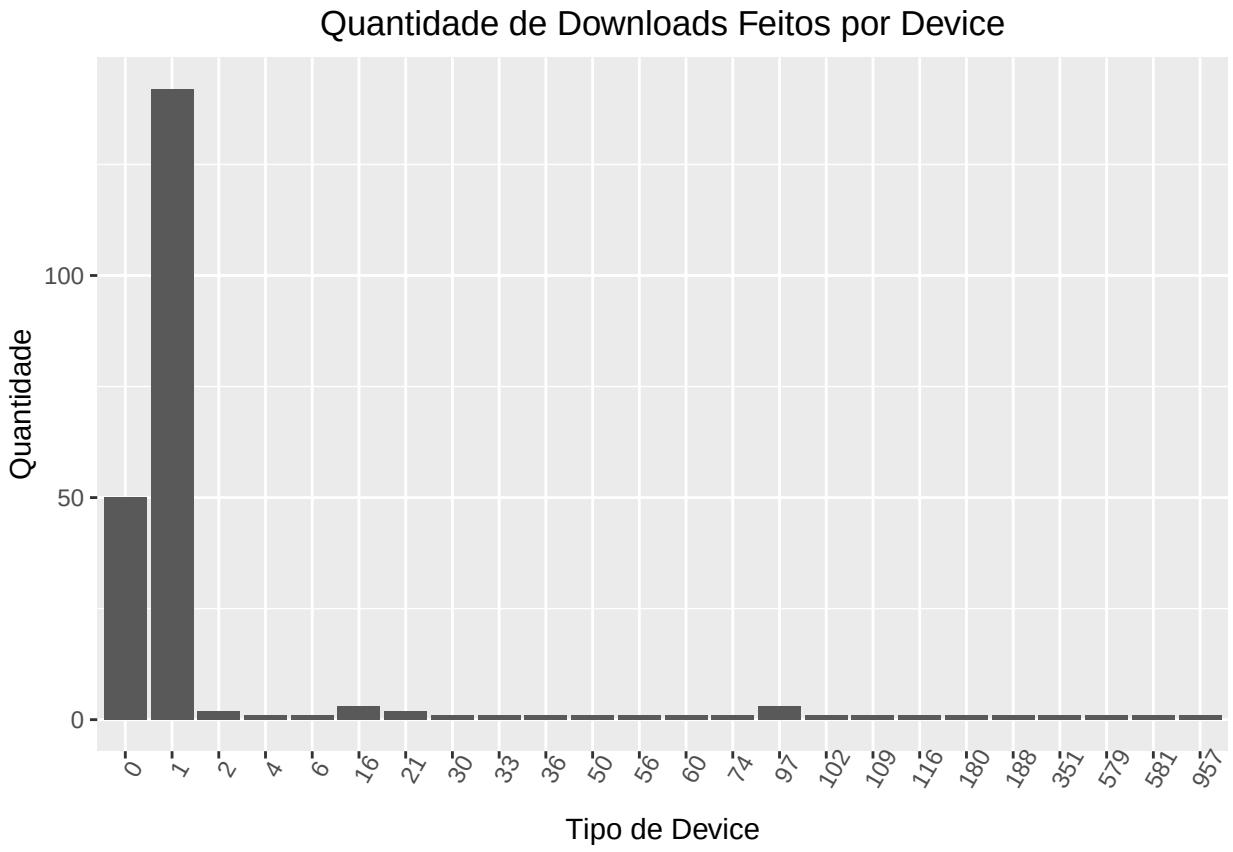
```
nrow(df[df$app==35])
```

```
## [1] 49
```

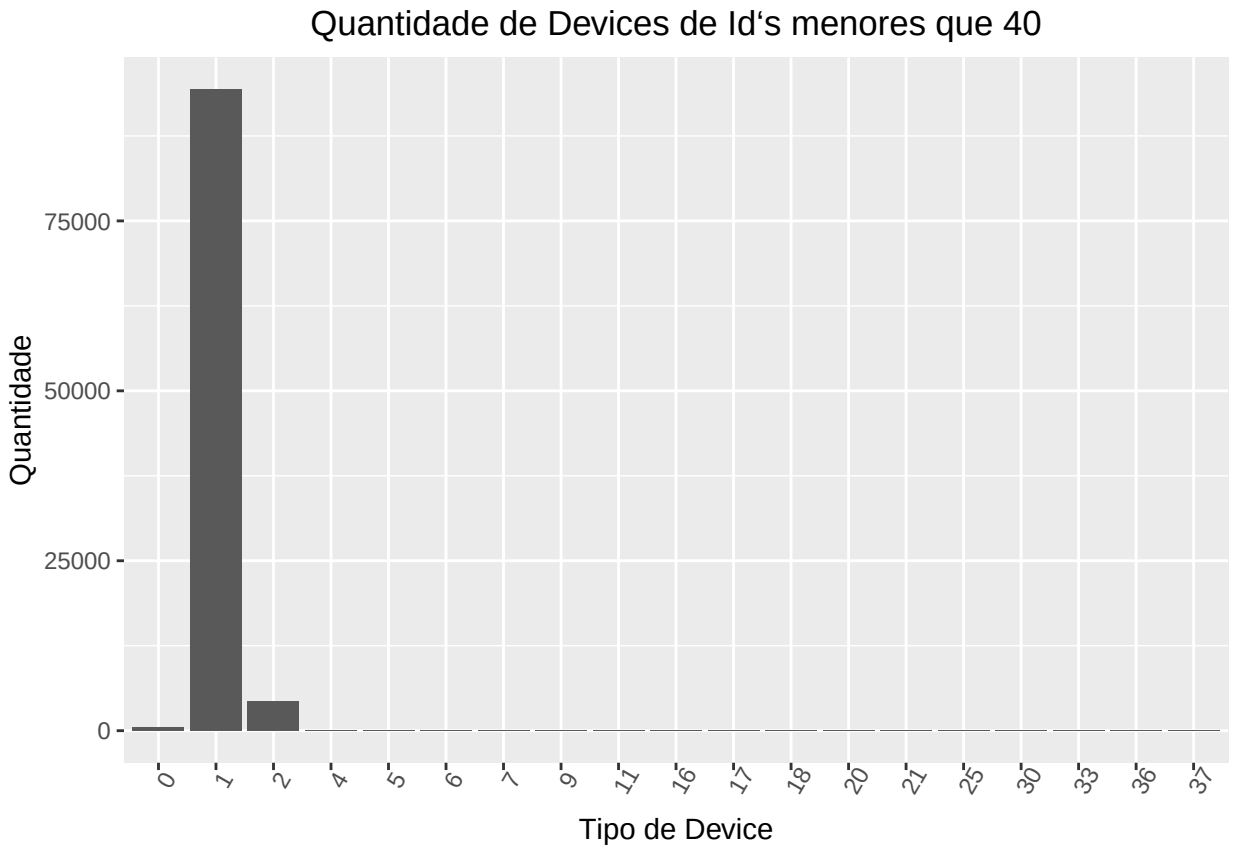
```
nrow(df[df$app==35 & df$is_attributed ==1])
```

```
## [1] 27
```

```
# Vendo em números podemos ver até que os apps do tipo 35 têm mais downloads do que apenas cliques.

# Analizando a variável device
df %>%
  filter(is_attributed == 1) %>%
  ggplot(aes(x=as.factor(device), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Downloads Feitos por Device") +
    xlab("Tipo de Device") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```




```
df %>%
  filter(device < 40) %>%
  ggplot(aes(as.factor(device))) +
    geom_bar(stat="count") +
    ggtitle("Quantidade de Devices de Id`s menores que 40") +
    xlab("Tipo de Device") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```




```
# Como podemos ver a maioria dos downloads foram feitos pelos devices 0 e 1, porém a quantidade de disp

# Número de devices do tipo 1
nrow(df[df$device==1])
```

```
## [1] 94334
```

```
# Número de downloads do device do tipo 1
nrow(df[df$device==1 & df$is_attributed ==1])
```

```
## [1] 142
```

```
# Proporção de downloads feitos de device 1
nrow(df[df$device==1 & df$is_attributed ==1]) /
  (nrow(df[df$device==1]) + nrow(df[df$device==1 & df$is_attributed ==1]))
```

```
## [1] 0.001503027
```

```
# Número de devices do tipo 0
nrow(df[df$device==0])
```

```
## [1] 539
```

```
# Número de downloads do device do tipo 0
nrow(df[df$device==0 & df$is_attributed ==1])
```

```
## [1] 50
```

```
# Proporção de downloads feitos de device 0
nrow(df[df$device==0 & df$is_attributed ==1]) /
  (nrow(df[df$device==0]) + nrow(df[df$device==0 & df$is_attributed ==1]))
```

```
## [1] 0.08488964
```

```
# Número de devices do tipo 16
nrow(df[df$device==16])
```

```
## [1] 7
```

```
# Número de downloads do device do tipo 16
nrow(df[df$device==16 & df$is_attributed ==1])
```

```
## [1] 3
```

```
# Proporção de downloads feitos de device 16
nrow(df[df$device==16 & df$is_attributed ==1]) /
  (nrow(df[df$device==16]) + nrow(df[df$device==16 & df$is_attributed ==1]))
```

```
## [1] 0.3
```

```
# Número de devices do tipo 97
nrow(df[df$device==97])
```

```
## [1] 5
```

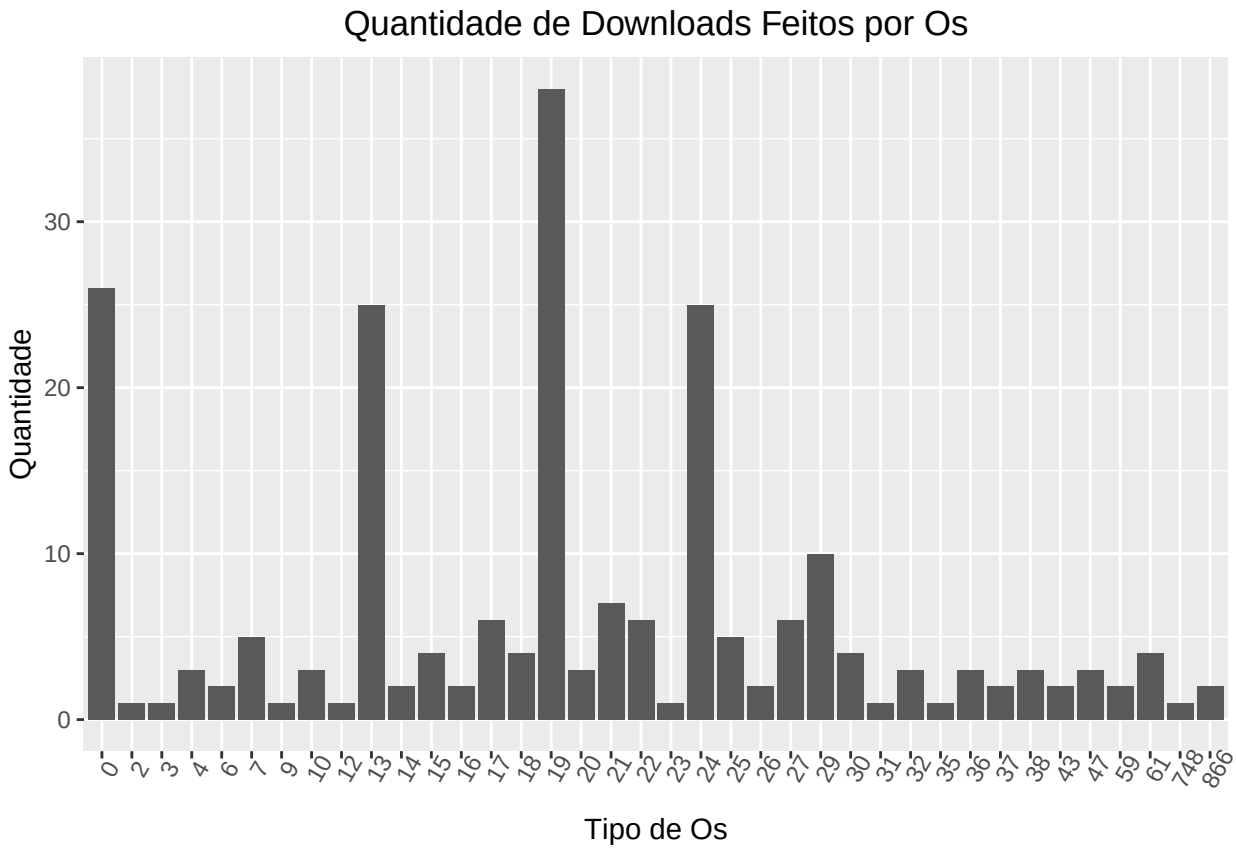
```
# Número de downloads do device do tipo 97
nrow(df[df$device==97 & df$is_attributed ==1])
```

```
## [1] 3
```

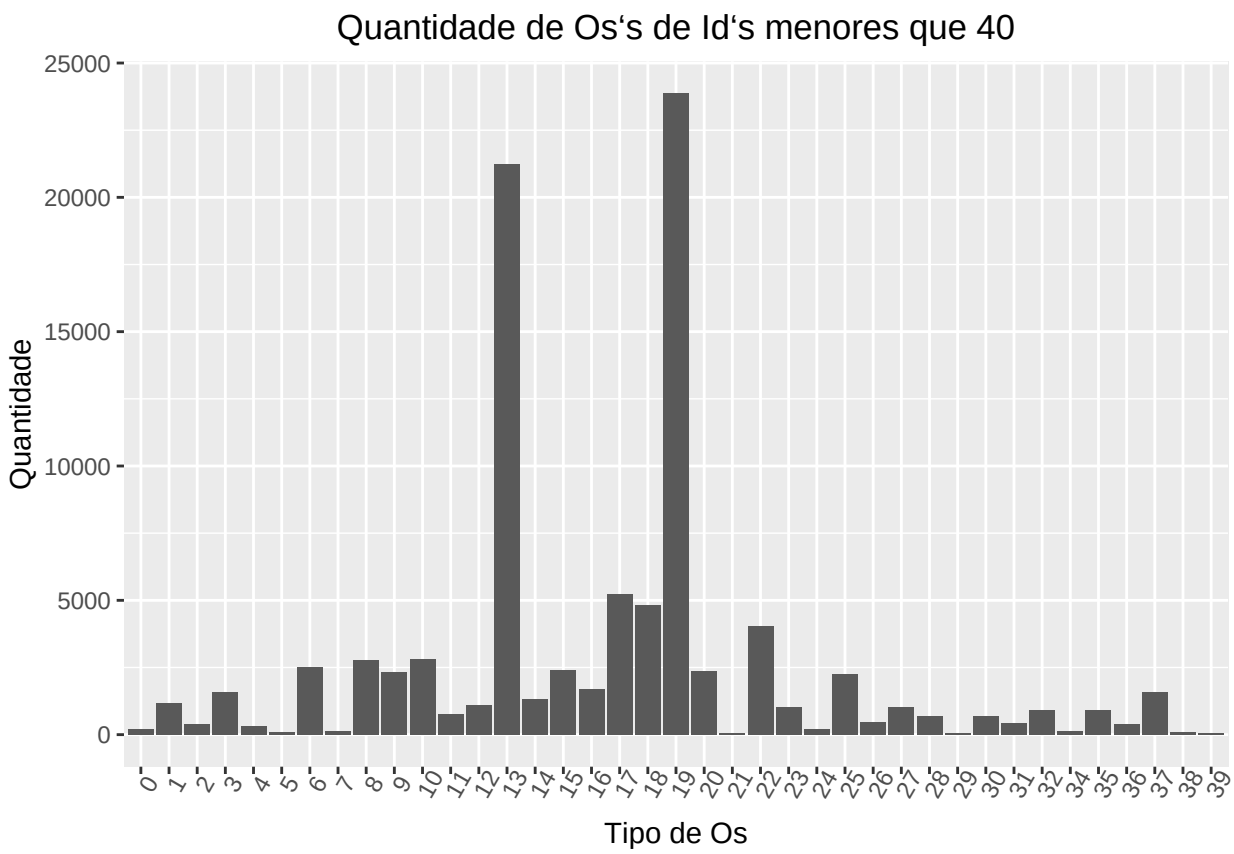
```
# Proporção de downloads feitos de device 97
nrow(df[df$device==97 & df$is_attributed ==1]) /
  (nrow(df[df$device==97]) + nrow(df[df$device==97 & df$is_attributed ==1]))
```

```
## [1] 0.375
```

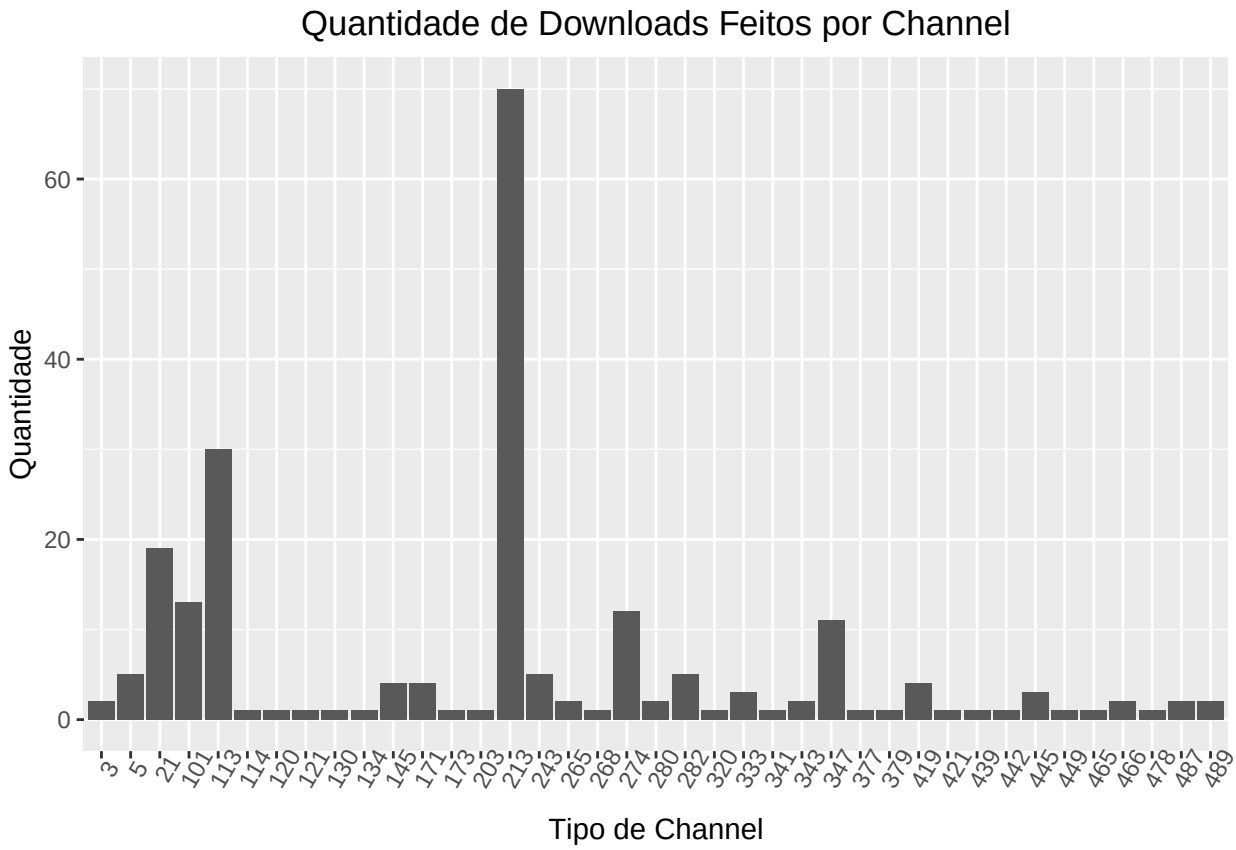
```
# Analizando os devices do tipo 16 e 97 percebemos que eles têm uma proporção bem maior de downloads do

# Analizando a variável OS
df %>%
  filter(is_attributed == 1) %>%
  ggplot(aes(x=as.factor(os), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Downloads Feitos por Os") +
    xlab("Tipo de Os") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```



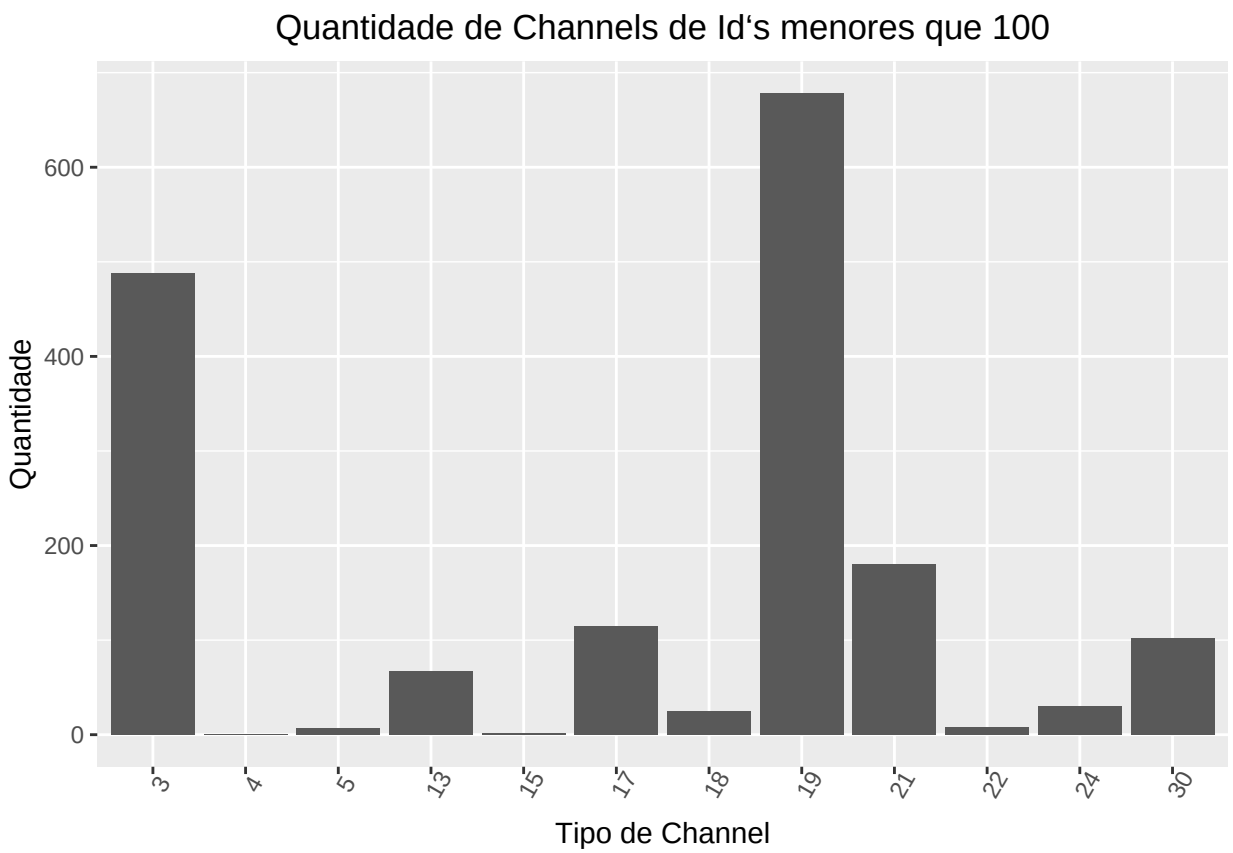

```
df %>%
  filter(os < 40) %>%
  ggplot(aes(as.factor(os), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Os`s de Id`s menores que 40") +
    xlab("Tipo de Os") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```



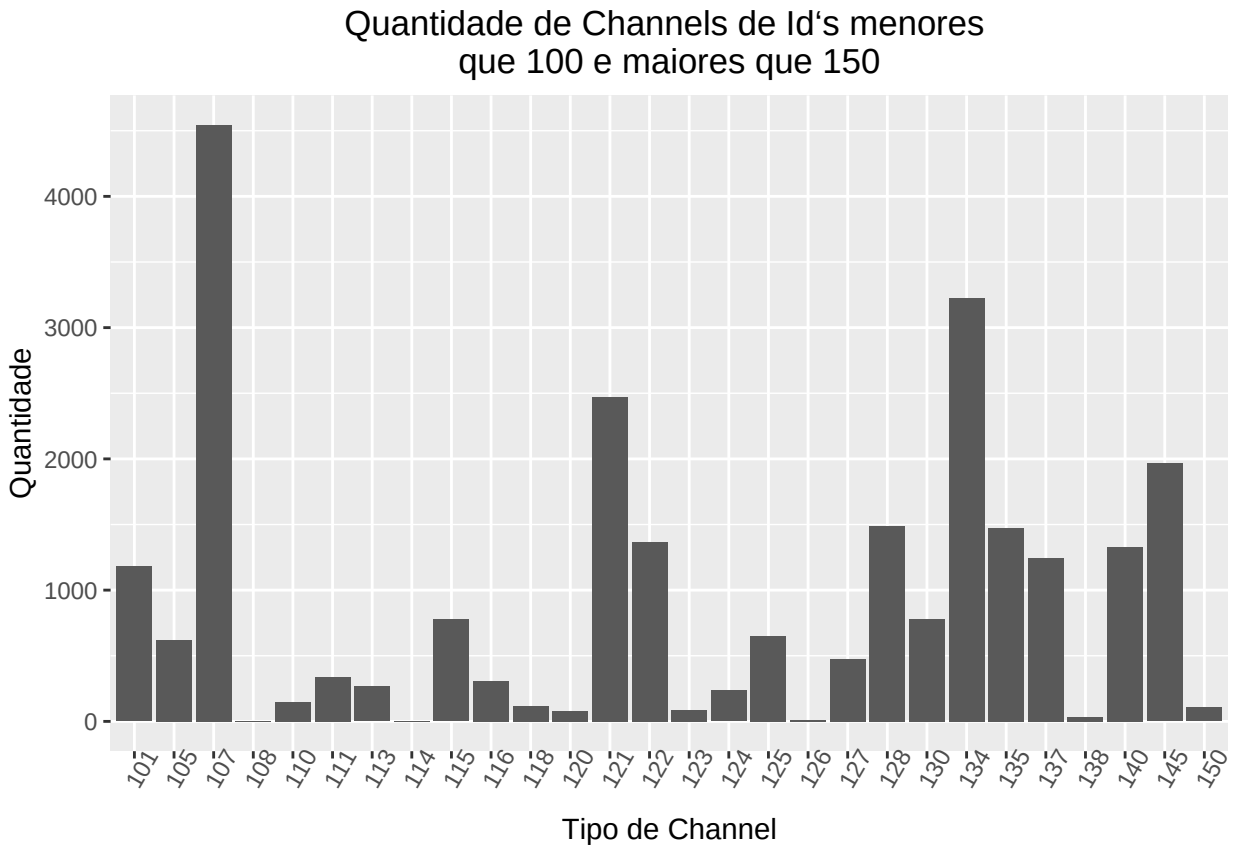

```
# Podemos ver que temos muitos os`s do tipo 19, 13, 0 e 24 que fizeram download do app, porém os do tip

# Analizando a variável channel
df %>%
  filter(is_attributed == 1) %>%
  ggplot(aes(x=as.factor(channel), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Downloads Feitos por Channel") +
    xlab("Tipo de Channel") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```



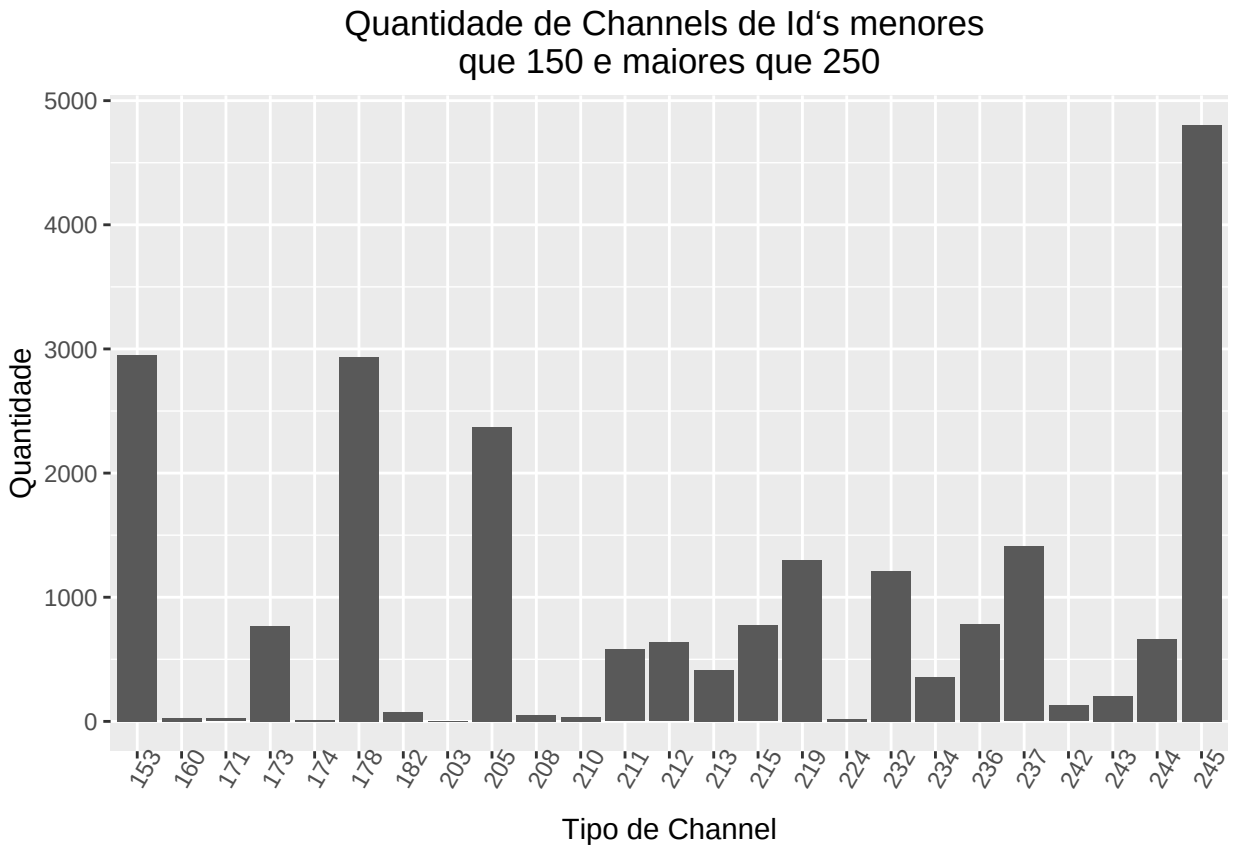

```
df %>%
  filter(channel <= 100) %>%
  ggplot(aes(as.factor(channel), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Channels de Id`s menores que 100") +
    xlab("Tipo de Channel") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```



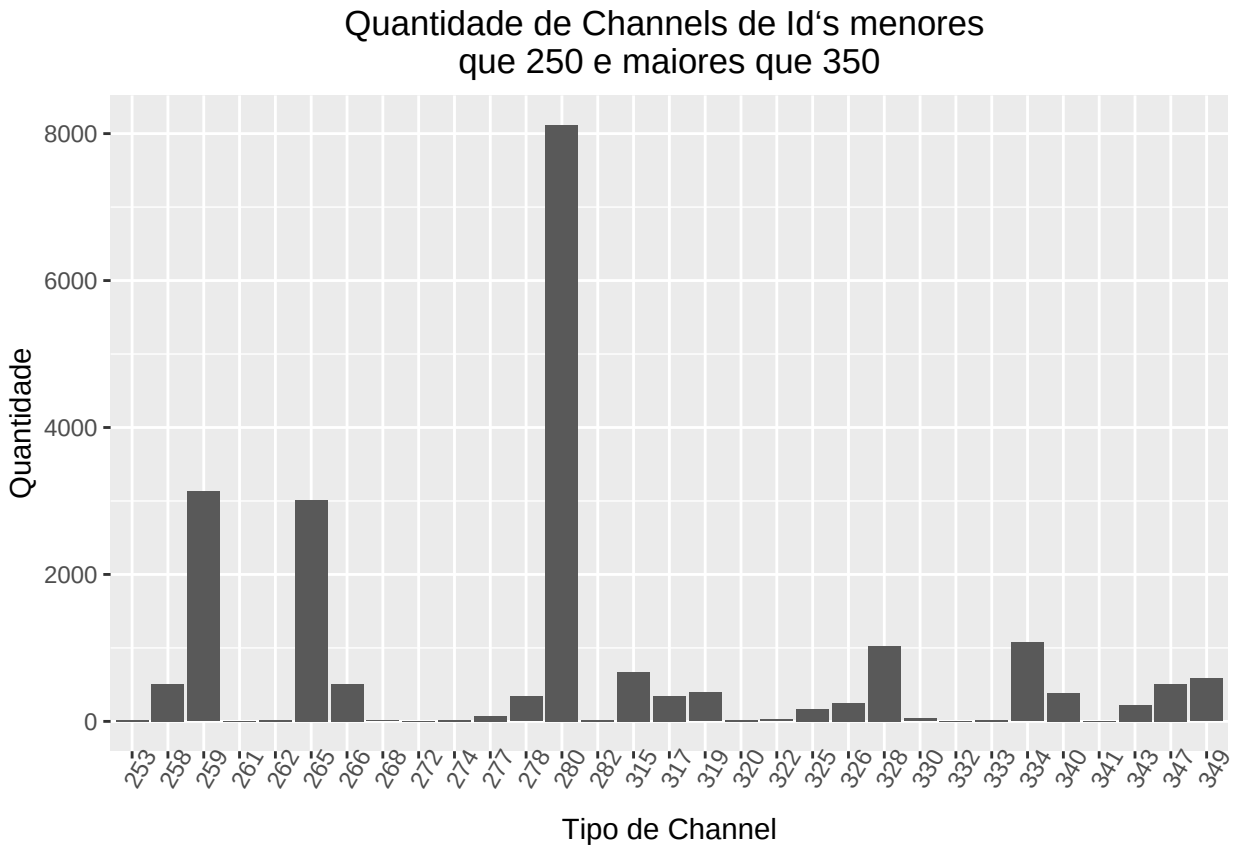

```
df %>%
  filter(channel > 100 & channel <= 150) %>%
  ggplot(aes(as.factor(channel), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Channels de Id`s menores \nque 100 e maiores que 150") +
    xlab("Tipo de Channel") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```



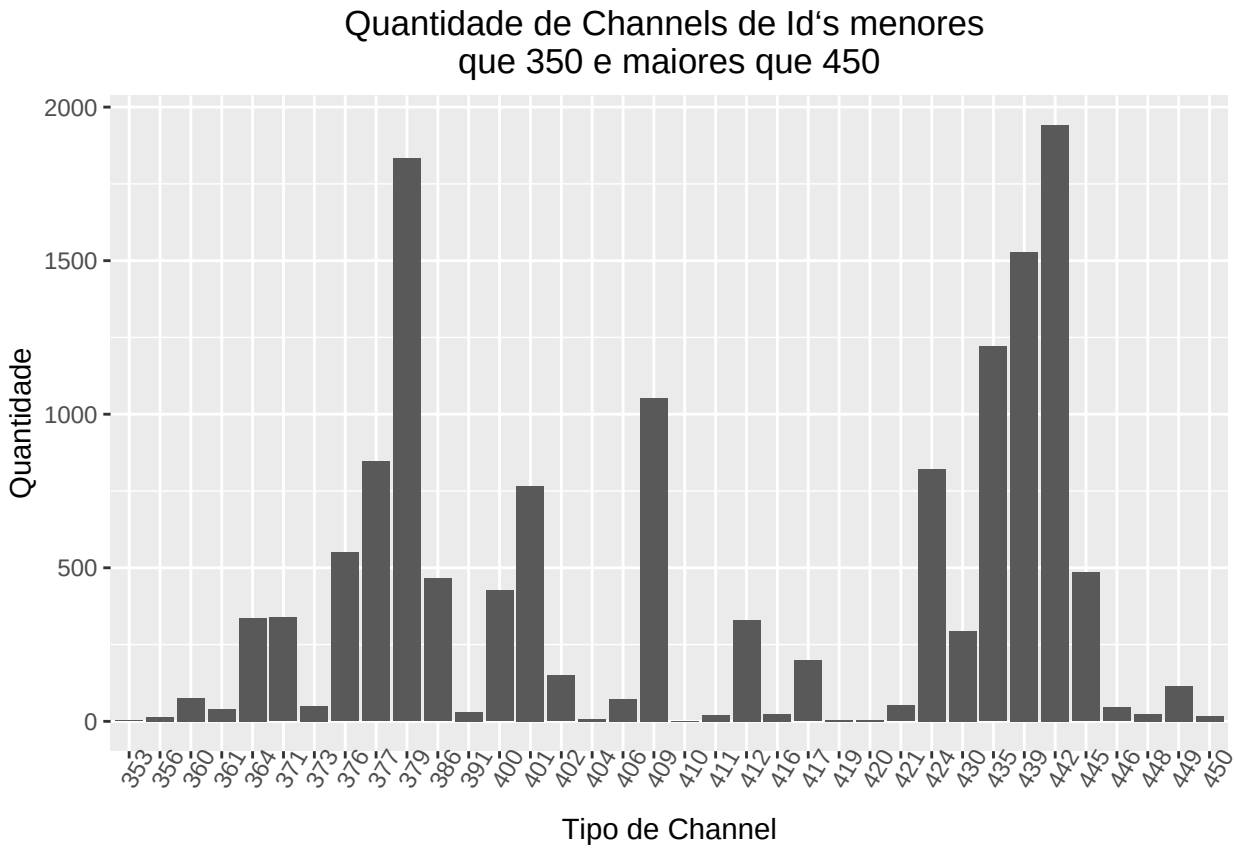

```
df %>%
  filter(channel > 150 & channel <= 250) %>%
  ggplot(aes(as.factor(channel), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Channels de Id`s menores \nque 150 e maiores que 250") +
    xlab("Tipo de Channel") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```



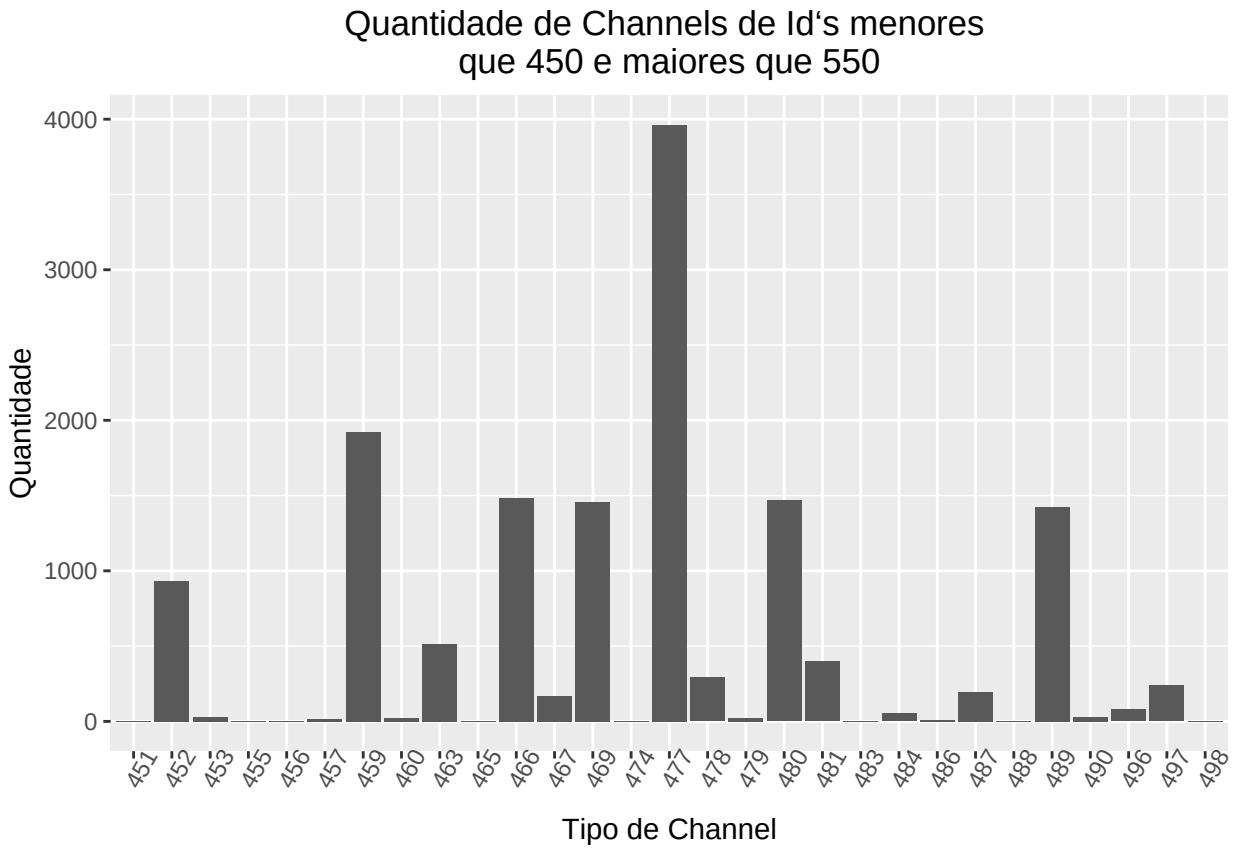

```
df %>%
  filter(channel > 250 & channel <= 350) %>%
  ggplot(aes(as.factor(channel), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Channels de Id`s menores \nque 250 e maiores que 350") +
    xlab("Tipo de Channel") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```




```
df %>%
  filter(channel > 350 & channel <= 450) %>%
  ggplot(aes(as.factor(channel), ..count..)) +
    geom_bar() +
    ggtitle("Quantidade de Channels de Id`s menores \nque 350 e maiores que 450") +
    xlab("Tipo de Channel") +
    ylab("Quantidade") +
    theme(axis.text.x = element_text(angle = 60),
          plot.title = element_text(hjust = 0.5))
```



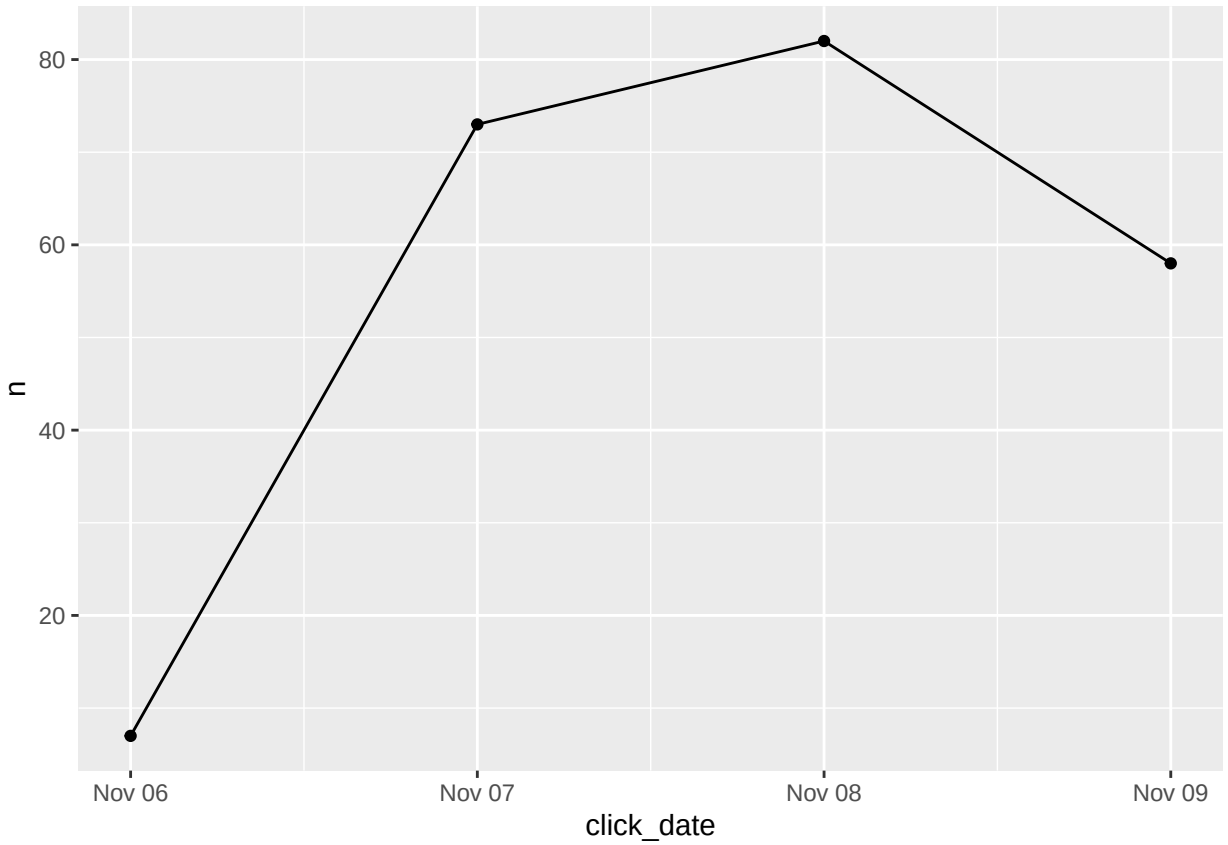

```
df %>%
  filter(channel > 450 & channel <= 550) %>%
  ggplot(aes(as.factor(channel), ..count..)) +
  geom_bar() +
  ggtitle("Quantidade de Channels de Id`s menores \nque 450 e maiores que 550") +
  xlab("Tipo de Channel") +
  ylab("Quantidade") +
  theme(axis.text.x = element_text(angle = 60),
        plot.title = element_text(hjust = 0.5))
```



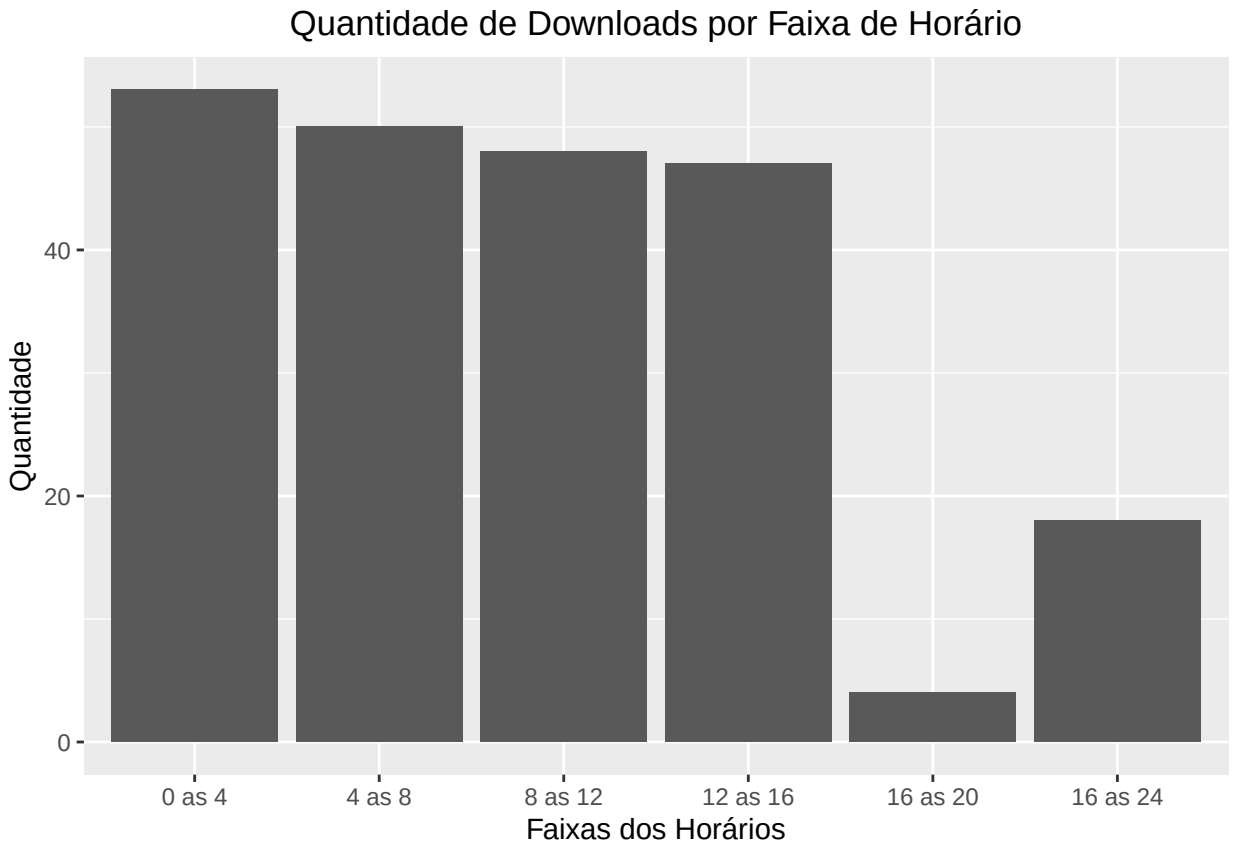

```
# Podemos ver que alguns channels são bem acessados porém exitem poucos downloads por eles, por outro l

# Analizando a variável click date
table(df$click_date)
```

```
##
## 2017-11-06 2017-11-07 2017-11-08 2017-11-09
##       5010      32389      34034      28560
```

```
df %>%
  filter(is_attributed == 1) %>%
  count(click_date) %>%
  ggplot(aes(x = click_date, n)) +
  geom_point() +
  geom_line()
```




```
# A data não me parece influenciar muito no download do app, mas estou curioso com relação ao horário,
# Função para converter horas em segundos
convert_hour <- function(hour){
  h <- 60*60*hour
  return(h)
}
# Criando a variável range_hour que separa o dia em 6 partes (4 em 4 horas).
df$range_hour <-findInterval(df$click_hour,c(convert_hour(0), convert_hour(4),
                                             convert_hour(8), convert_hour(12),
                                             convert_hour(16), convert_hour(20),
                                             convert_hour(24)))
df$range_hour = as.factor(df$range_hour)
levels(df$range_hour)<-c("0 as 4", "4 as 8", "8 as 12", "12 as 16", "16 as 20", "16 as 24")

df %>%
  filter(is_attributed == 1) %>%
  ggplot(aes(x = range_hour, ..count..)) +
  geom_bar() +
  ggtitle("Quantidade de Downloads por Faixa de Horário") +
  xlab("Faixas dos Horários") +
  ylab("Quantidade") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Podemos ver que a maior parte dos downloads são feitos de madrugada, entre meia noite e 04 da manhã e
```

Primeiro Modelo Preditivo

Para o nosso primeiro modelo preditivo vou considerar as colunas app, device, os, channel count_ip e range_hour. Esse modelo terá poucas alterações com relação ao arquivo de amostra, pois assim teremos uma baseline para melhorar nas próximas previsões.

```
# Fazendo o split do data set para treinar o modelo.
require(caret)
```

```
## Loading required package: caret
```

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

```
set.seed(123)
trainIndex <- createDataPartition(df$is_attributed, p = 0.7,list = FALSE,  times = 1)
train <- df[trainIndex,]
test <- df[-trainIndex,]

# Criando dos modelos preditivos
formula_v1 <- as.formula('is_attributed ~ app + device + os + channel + count_ip + range_hour')
# Treinando o modelo com o algoritmo de regressão logística
model_glm_v1 <- glm(formula = formula_v1, data = train, family = "binomial")
```

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

```
# Verificando alguns resultados do modelo treinado
summary(model_glm_v1)
```

```
##
## Call:
## glm(formula = formula_v1, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1882  -0.0783  -0.0541  -0.0298   5.5253
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -4.6525348  0.2472354 -18.818  < 2e-16 ***
## app                 0.0197239  0.0019143  10.304  < 2e-16 ***
## device             -0.0015018  0.0005854  -2.565  0.01031 *
## os                  0.0014713  0.0016274   0.904  0.36593
## channel            -0.0041059  0.0007119  -5.768 8.04e-09 ***
## count_ip           -0.1440773  0.0262884  -5.481 4.24e-08 ***
## range_hour4 as 8   -0.0631296  0.2360734  -0.267  0.78915
## range_hour8 as 12  -0.0824561  0.2421580  -0.341  0.73348
## range_hour12 as 16  0.0180216  0.2422436   0.074  0.94070
## range_hour16 as 20 -2.4398282  0.9386248  -2.599  0.00934 **
## range_hour16 as 24  0.0971521  0.3188418   0.305  0.76059
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2192.4  on 69995  degrees of freedom
## Residual deviance: 1980.3  on 69985  degrees of freedom
## AIC: 2002.3
##
## Number of Fisher Scoring iterations: 13
```

```r
# Realizando a predição com o modelo treinado
pred_glm_v1 <- predict(model_glm_v1, test, type="response")
# Arredondando para 0 ou 1
pred_glm_v1 <- round(pred_glm_v1)
#Confusion Matrix da predição.
confusionMatrix(table(data = pred_glm_v1, reference = test$is_attributed),
                positive = '1')
```

```
## Confusion Matrix and Statistics
##
##     reference
## data     0     1
##    0 29927    66
##    1     4     0
##
##                Accuracy : 0.9977
##                  95% CI : (0.9971, 0.9982)
##     No Information Rate : 0.9978
##     P-Value [Acc > NIR] : 0.7152
##
##                   Kappa : -3e-04
##
##  Mcnemar's Test P-Value : 3.079e-13
```

```
##
##             Sensitivity : 0.0000000
##             Specificity : 0.9998664
##          Pos Pred Value : 0.0000000
##          Neg Pred Value : 0.9977995
##              Prevalence : 0.0022002
##          Detection Rate : 0.0000000
##    Detection Prevalence : 0.0001333
##       Balanced Accuracy : 0.4999332
##
##        'Positive' Class : 1
##
```

```
# Curva roc para o model_glm_v1
#install.packages("plotROC")
library(ROCR)
library(pROC)
```

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

```
##
## Attaching package: 'pROC'
```

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

```
library(ROSE)
```

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

```
auc(test$is_attributed,  pred_glm_v1)
```

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

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.4999
```

```
roc.curve(test$is_attributed, pred_glm_v1, plotit = T, col = "red")
```

```
## Area under the curve (AUC): 0.500
```

```
# Nosso modelo teve uma ótima acurácia de 99%, mas a pela métrica AUC o desempenho
# não foi tão bom assim. Vamo tester outros algoritmos e depois fazer o balanceamento
# da variável is_target.

# Criando o modelo com o algoritmo Árvore de Decissão
library(C50)
# Treinando o modelo
modelo_tree_v1 = C5.0(formula_v1, data = train)
# Previsões nos dados de teste
pred_tree_v1 = predict(modelo_tree_v1, test, type='class')
# Confusion Matrix
confusionMatrix(test$is_attributed, pred_tree_v1, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
```

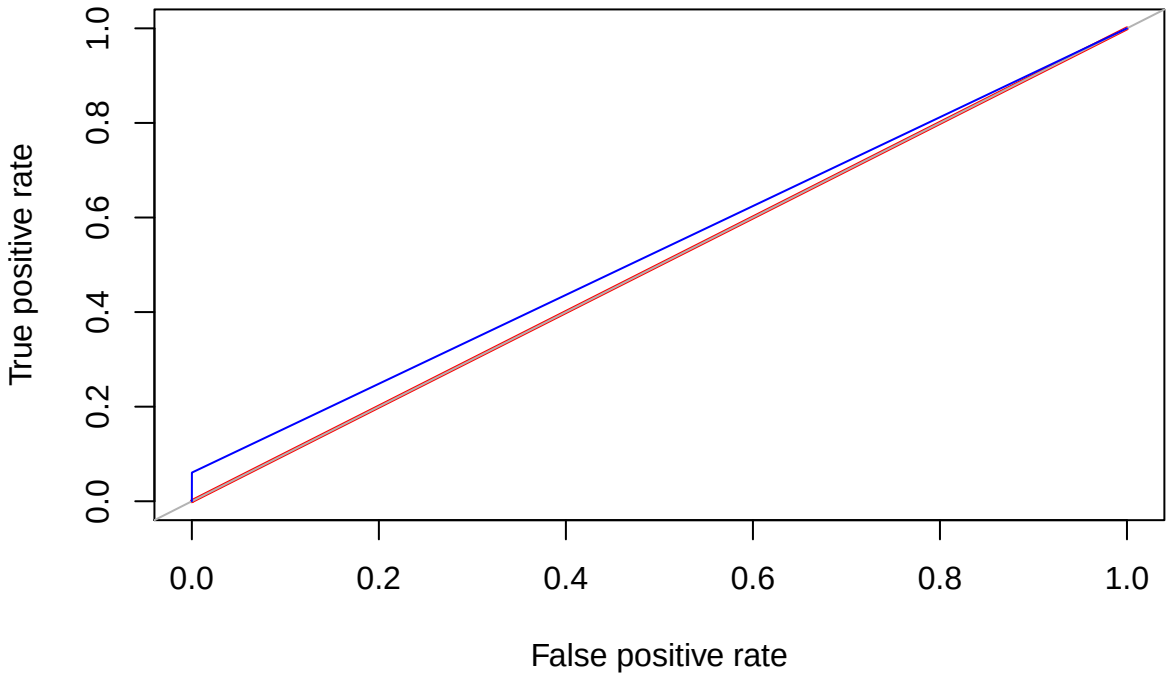
```
##          0 29929     2
##          1    62     4
##
##                Accuracy : 0.9979
##                  95% CI : (0.9973, 0.9984)
##     No Information Rate : 0.9998
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1108
##
##  Mcnemar's Test P-Value : 1.643e-13
##
##             Sensitivity : 0.6666667
##             Specificity : 0.9979327
##          Pos Pred Value : 0.0606061
##          Neg Pred Value : 0.9999332
##              Prevalence : 0.0002000
##          Detection Rate : 0.0001333
##    Detection Prevalence : 0.0022002
##       Balanced Accuracy : 0.8322997
##
##        'Positive' Class : 1
##
```

```
pred_tree_v1 <- as.numeric(pred_tree_v1)
auc(test$is_attributed, as.numeric(pred_tree_v1))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
## Area under the curve: 0.5303
```

```
roc.curve(test$is_attributed, pred_tree_v1, plotit = T, col = "blue", add.roc = T)
```




```
## Area under the curve (AUC): 0.530
```

```r
# Tanto nesse modelo como no anterior podemos ver que o alogritmo aprendeu muito
# sobre o evento de não se fazer download, porém também erra muito ao tentar descobrir
# se a pessoa fez o downaload. O fato da acuracia ser alta não significa que
# o modelo esteja prevendo direito, apenas que ele acerta muito porque mais de 99%
# dos casos são de downloads não realizados. Por isso o balanceamento é importante.

# Antes vou submeter o modelo de regressão linear no kaggle para ter uma base para melhoria
# do algoritmo.
# Como o arquivo de teste é muito grande, estava ocupando muita memória em meu computador,
# então decidi particionar o teste em 4 partes que foram gravados em arquivos csv, e agora
# vou ler eles em separado.

# função para transformar os dados do arquivo teste
transform_test <- function(test){
  test$click_time2 <- test$click_time
  test <- test %>%
    separate(click_time2, c("click_date", "click_hour"), " ")
  test$click_hour <- as_hms(test$click_hour)
  test$ip <- as.character(test$ip)
  test$count_ip <- as.numeric(ave(test$ip, test$ip, FUN = length))
  test$range_hour <-findInterval(test$click_hour, c(convert_hour(0), convert_hour(4), convert_hour(8),
  test$range_hour = as.factor(test$range_hour)
  levels(test$range_hour)<-c("0 as 4", "4 as 8", "8 as 12", "12 as 16", "16 as 20", "16 as 24")
  return(test)
}

# Lendo o arquivo test_p1 e fazendo a previsão com o agloritmo glm
test_kaggle_p1 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
```

```r
# transformando o arquivo para a predição
test_kaggle_p1 <- transform_test(test_kaggle_p1)
# realizando a predição
pred_glm_v1_p1 <- predict(model_glm_v1, test_kaggle_p1, type="response")
# arredondando os valores preditos
pred_glm_v1_p1 <- round(pred_glm_v1_p1)
# criando um data frame com resposta e predição
prediction1_p1 <- data.frame('click_id' = test_kaggle_p1$click_id,
                             'is_attributed' = pred_glm_v1_p1)
# Visualizando o data frame
head(prediction1_p1)
```

```
##   click_id is_attributed
## 1        0             0
## 2        1             0
## 3        2             0
## 4        3             0
## 5        4             0
## 6        5             0
```

```r
# Excluindo o arquivo de teste
rm(test_kaggle_p1)

# Lendo o arquivo test_p2 e fazendo a previsão com o agloritmo glm
test_kaggle_p2 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
# transformando o arquivo para a predição
test_kaggle_p2 <- transform_test(test_kaggle_p2)
# realizando a predição
head(test_kaggle_p2)
```

```
##    click_id     ip app device os channel          click_time click_date
## 1:  4697617 117122   2      1 19     477 2017-11-10 05:27:21 2017-11-10
## 2:  4697618   5348  17      1 13     128 2017-11-10 05:27:21 2017-11-10
## 3:  4697619   8366  14      1 19     480 2017-11-10 05:27:21 2017-11-10
## 4:  4697620  71710   9      1 13     232 2017-11-10 05:27:21 2017-11-10
## 5:  4697621   5250  18      1 13     134 2017-11-10 05:27:21 2017-11-10
## 6:  4697622  87879   1      1 19     134 2017-11-10 05:27:21 2017-11-10
##    click_hour count_ip range_hour
## 1:   05:27:21      123     0 as 4
## 2:   05:27:21    40200     0 as 4
## 3:   05:27:21      131     0 as 4
## 4:   05:27:21      297     0 as 4
## 5:   05:27:21      361     0 as 4
## 6:   05:27:21     1318     0 as 4
```

```r
# arredondando os valores preditos
pred_glm_v1_p2 <- predict(model_glm_v1, test_kaggle_p2, type="response")
# criando um data frame com resposta e predição
pred_glm_v1_p2 <- round(pred_glm_v1_p2)
# criando um data frame com resposta e predição
prediction1_p2 <- data.frame('click_id' = test_kaggle_p2$click_id,
                             'is_attributed' = pred_glm_v1_p2)
# Visualizando o data frame
head(prediction1_p2)
```

```
##   click_id is_attributed
## 1  4697617             0
## 2  4697618             0
## 3  4697619             0
## 4  4697620             0
## 5  4697621             0
## 6  4697622             0
```

```
# Excluindo o arquivo de teste
rm(test_kaggle_p2)

# Lendo o arquivo test_p3 e fazendo a previsão com o agloritmo glm
test_kaggle_p3 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
# transformando o arquivo para a predição
test_kaggle_p3 <- transform_test(test_kaggle_p3)
# realizando a predição
head(test_kaggle_p3)
```

```
##    click_id     ip app device os channel          click_time click_date
## 1:  9395233  69447  12      1 14     140 2017-11-10 10:03:52 2017-11-10
## 2:  9395235  99024   2      1  3     122 2017-11-10 10:03:52 2017-11-10
## 3:  9395236 124976  14      1 13     349 2017-11-10 10:03:52 2017-11-10
## 4:  9395237 108838   9      1 41     215 2017-11-10 10:03:52 2017-11-10
## 5:  9395238 118229  11      1 18     219 2017-11-10 10:03:52 2017-11-10
## 6:  9395239  39889  14      1 19     442 2017-11-10 10:03:52 2017-11-10
##    click_hour count_ip range_hour
## 1:   10:03:52      388     0 as 4
## 2:   10:03:52      374     0 as 4
## 3:   10:03:52      285     0 as 4
## 4:   10:03:52       61     0 as 4
## 5:   10:03:52     2364     0 as 4
## 6:   10:03:52      275     0 as 4
```

```
# arredondando os valores preditos
pred_glm_v1_p3 <- predict(model_glm_v1, test_kaggle_p3, type="response")
# criando um data frame com resposta e predição
pred_glm_v1_p3 <- round(pred_glm_v1_p3)
# criando um data frame com resposta e predição
prediction1_p3 <- data.frame('click_id' = test_kaggle_p3$click_id,
                             'is_attributed' = pred_glm_v1_p3)
# Visualizando o data frame
head(prediction1_p3)
```

```
##   click_id is_attributed
## 1  9395233             0
## 2  9395235             0
## 3  9395236             0
## 4  9395237             0
## 5  9395238             0
## 6  9395239             0
```

```
# Excluindo o arquivo de teste
rm(test_kaggle_p3)

# Lendo o arquivo test_p4 e fazendo a previsão com o agloritmo glm
test_kaggle_p4 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
```

```
# transformando o arquivo para a predição
test_kaggle_p4 <- transform_test(test_kaggle_p4)
# realizando a predição
head(test_kaggle_p4)
```

```
##    click_id     ip app device os channel          click_time click_date
## 1: 14092851  88718  15      1 28     430 2017-11-10 13:34:07 2017-11-10
## 2: 14092852  24181  12      1 19     178 2017-11-10 13:34:07 2017-11-10
## 3: 14092853  65380   9      1 19     215 2017-11-10 13:34:07 2017-11-10
## 4: 14092854   1946  11      1 13     319 2017-11-10 13:34:07 2017-11-10
## 5: 14092855  90850   1      2  9     125 2017-11-10 13:34:07 2017-11-10
## 6: 14092856 106035   9      2 13     258 2017-11-10 13:34:07 2017-11-10
##    click_hour count_ip range_hour
## 1:   13:34:07      186     0 as 4
## 2:   13:34:07      120     0 as 4
## 3:   13:34:07       19     0 as 4
## 4:   13:34:07       21     0 as 4
## 5:   13:34:07     9102     0 as 4
## 6:   13:34:07      850     0 as 4
```

```
# arredondando os valores preditos
pred_glm_v1_p4 <- predict(model_glm_v1, test_kaggle_p4, type="response")
# criando um data frame com resposta e predição
pred_glm_v1_p4 <- round(pred_glm_v1_p4)
# criando um data frame com resposta e predição
prediction1_p4 <- data.frame('click_id' = test_kaggle_p4$click_id,
                             'is_attributed' = pred_glm_v1_p4)
# Visualizando o data frame
head(prediction1_p4)
```

```
##   click_id is_attributed
## 1 14092851             0
## 2 14092852             0
## 3 14092853             0
## 4 14092854             0
## 5 14092855             0
## 6 14092856             0
```

```
# Excluindo o arquivo de teste
rm(test_kaggle_p4)

# Juntando as 4 previsões
prediction1 <- rbind(prediction1_p1, prediction1_p2,
                     prediction1_p3, prediction1_p4)

# Escrevendo o arquivo em formato csv.
#fwrite(prediction1, "prediction1.csv", row.names = F, sep = ",")

# Removendo alguns dados para limpar a memória do R
rm(prediction1_p1, prediction1_p2, prediction1_p3, prediction1_p4)
rm(pred_glm_v1_p1, pred_glm_v1_p2, pred_glm_v1_p3, pred_glm_v1_p4)
rm(pred_glm_v1, pred_tree_v1)
rm(prediction1)
rm(model_glm_v1, modelo_tree_v1)
```

```
# Essa predição deu como resultado 0.5 na curva AUC, o que é um resultado muito
# ruim. Vou fazer o balanceamento para ver como fica o resultado.
```

Segundo Modelo Preditivo

Como previsto, a medida AUC ficou muito ruim no primeiro modelo, pois como os dados estão distribuídos de forma desigual, 99% não e 1% sim, o algoritmo aprende muito com relação ao evento não, mas pouco em relação ao sim. Por isso o balanceamente é importante, pois assim o modelo consegue aprender tanto para o evente sim como para o não da mesma forma. Existem diversas maneiras de se balancear data set, podemos apenas tirar vários dados de forma que eles fiquem iguais e isso é bom pelo ponto de vista dos dados continuarem sendo reais, mas por outro lado perdemos muitos dados e o algoritmo aprenderá pouco. Também podemos criar dados através de uma estimativa condicional dos dados que queremos inserir.

Neste segundo modelo preditivo, vamos utilizar a função ROSE que cria uma amostra de dados sintéticos, ampliando o espaço de recursos de exemplos de classe minoritária e majoritária. Assim, teremos um data set mais balanceado na modelagem preditiva.

```
# Balanceando os dados
# Feature selection nos dados de treino e teste
train <- train %>% select(app, device, os, channel, count_ip,
                          range_hour, is_attributed)
test <- test %>% select(app, device, os, channel, count_ip,
                        range_hour, is_attributed)
# ROSE nos dados de treino
rose_train <- ROSE(is_attributed ~ ., data = train, seed = 1)$data
prop.table(table(rose_train$is_attributed))
```

```
##
##         0         1
## 0.4988142 0.5011858
```

```
# ROSE nos dados de teste
rose_test <- ROSE(is_attributed ~ ., data = test, seed = 1)$data
prop.table(table(rose_test$is_attributed))
```

```
##
##         0         1
## 0.5040171 0.4959829
```

```
# Agora vamos criar o modelo preditivo.
# Treinando o modelo com o algoritmo de regressão logística
model_glm_v2 <- glm(is_attributed ~ ., data = rose_train, family = "binomial")
# Verificando alguns resultados do modelo treinado
summary(model_glm_v2)
```

```
##
## Call:
## glm(formula = is_attributed ~ ., family = "binomial", data = rose_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.1350  -0.9757   0.0207   0.9670   3.0157
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.997e-01  2.573e-02   7.762 8.37e-15 ***
```

```
## app                 4.016e-02  5.233e-04  76.750  < 2e-16 ***
## device             -5.065e-05  5.077e-05  -0.998   0.3185
## os                 -1.135e-03  1.753e-04  -6.474 9.56e-11 ***
## channel            -3.008e-03  6.589e-05 -45.648  < 2e-16 ***
## count_ip           -1.568e-02  3.889e-04 -40.310  < 2e-16 ***
## range_hour4 as 8   -3.971e-02  2.534e-02  -1.567   0.1171
## range_hour8 as 12   4.790e-02  2.548e-02   1.880   0.0601 .
## range_hour12 as 16 -1.375e-01  2.604e-02  -5.281 1.28e-07 ***
## range_hour16 as 20 -1.716e+00  5.684e-02 -30.193  < 2e-16 ***
## range_hour16 as 24  2.022e-01  3.492e-02   5.791 6.98e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 97035  on 69995  degrees of freedom
## Residual deviance: 78663  on 69985  degrees of freedom
## AIC: 78685
##
## Number of Fisher Scoring iterations: 6
```

```
# Realizando a predição com o modelo treinado
pred_glm_v2 <- predict(model_glm_v2, rose_test, type="response")
# Arredondando os valores
pred_glm_v2 <- round(pred_glm_v2)
#Confusion Matrix da predição.
confusionMatrix(table(data = pred_glm_v2, reference = rose_test$is_attributed),
                positive = '1')
```

```
## Confusion Matrix and Statistics
##
##     reference
## data     0     1
##    0 10810  5416
##    1  4309  9462
##
##                Accuracy : 0.6758
##                  95% CI : (0.6705, 0.6811)
##     No Information Rate : 0.504
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3512
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6360
##             Specificity : 0.7150
##          Pos Pred Value : 0.6871
##          Neg Pred Value : 0.6662
##              Prevalence : 0.4960
##          Detection Rate : 0.3154
##    Detection Prevalence : 0.4591
##       Balanced Accuracy : 0.6755
##
##        'Positive' Class : 1
```

```
##
```

```
# Curva roc para o model_glm_v2
roc.curve(rose_test$is_attributed, pred_glm_v2, plotit = T,
          col = "darkred")
```

```
## Area under the curve (AUC): 0.675
```

```
# A curva AUC teve uma boa melhora em relação ao algoritmo anterior, mas mesmo assim, continua fraca. V

# Criando o modelo com o algoritmo Árvore de Decissão
# Treinando o modelo
modelo_tree_v2 = C5.0(is_attributed ~ ., data = rose_train)
# Previsões nos dados de teste
pred_tree_v2 = predict(modelo_tree_v2, rose_test, type='class')
# Confusion Matrix
confusionMatrix(rose_test$is_attributed, pred_tree_v2, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13396  1723
##          1  1387 13491
##
##                Accuracy : 0.8963
##                  95% CI : (0.8928, 0.8998)
##     No Information Rate : 0.5072
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7927
##
##  Mcnemar's Test P-Value : 1.889e-09
##
##             Sensitivity : 0.8867
##             Specificity : 0.9062
##          Pos Pred Value : 0.9068
##          Neg Pred Value : 0.8860
##              Prevalence : 0.5072
##          Detection Rate : 0.4497
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.8965
##
##        'Positive' Class : 1
##
```

```
pred_tree_v2 <- as.numeric(pred_tree_v2)
# Para o modelo de árvore de decisão a acurácia não caiu tanto, mas agora vamos ver como a curva AUC se
# Curva roc para o modelo_tree_v2
roc.curve(rose_test$is_attributed, pred_tree_v2, plotit = T,
          col = "blue", add.roc = T)
```

```
## Area under the curve (AUC): 0.896
```

```
# Nossa curva AUC deu como resultado em 0.896, o que é excelente. Agora vamos tentar com outros algorit

# Criando o modelo com o algoritmo SVM (Suport Vector Machine)
```

```r
library(e1071)
# treinando o modelo
modelo_svm_v1 <- svm(is_attributed ~ ., data = rose_train,
                     type = 'C-classification', kernel = 'radial')
# Previsões nos dados de teste
pred_svm_v1 = predict(modelo_svm_v1, rose_test)
# Confusion Matrix
confusionMatrix(rose_test$is_attributed, pred_svm_v1, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13881  1238
##          1  2136 12742
##
##                Accuracy : 0.8875
##                  95% CI : (0.8839, 0.8911)
##     No Information Rate : 0.534
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7749
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9114
##             Specificity : 0.8666
##          Pos Pred Value : 0.8564
##          Neg Pred Value : 0.9181
##              Prevalence : 0.4660
##          Detection Rate : 0.4248
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.8890
##
##        'Positive' Class : 1
##
```

```r
# Curva roc para o modelo_svm_v1
roc.curve(rose_test$is_attributed, pred_svm_v1, plotit = T,
          col = "purple", add.roc = T)
```

```
## Area under the curve (AUC): 0.887
```

```r
# O desempenho tanto para árvore de decisão como para o SVM foi o mesmo, porém a árvore de decisão teve

# Criando o modelo com o algoritmo Random Forest
library(rpart)
# treinando o modelo
modelo_rf_v1 = rpart(is_attributed ~ ., data = rose_train,
                     control = rpart.control(cp = .0005))
# Previsões nos dados de teste
pred_rf_v1 = predict(modelo_rf_v1, rose_test, type='class')
# Confusion Matrix
confusionMatrix(rose_test$is_attributed, pred_rf_v1, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13149  1970
##          1   801 14077
##
##                Accuracy : 0.9076
##                  95% CI : (0.9043, 0.9109)
##     No Information Rate : 0.535
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8154
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8772
##             Specificity : 0.9426
##          Pos Pred Value : 0.9462
##          Neg Pred Value : 0.8697
##              Prevalence : 0.5350
##          Detection Rate : 0.4693
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9099
##
##        'Positive' Class : 1
##
```

```
# Curva roc para o modelo_rf_v1
roc.curve(rose_test$is_attributed, pred_rf_v1, plotit = T,
          col = "yellow", add.roc = T)
```

```
## Area under the curve (AUC): 0.908
```

```
# Criando o modelo com outro algoritmo Random Forest
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:ggplot2':
##
##     margin
```

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

```
# Treinando o modelo
modelo_rf_v2 = randomForest(formula = is_attributed ~ ., data = rose_train,
                            importance = TRUE)
# Previsões nos dados de teste
pred_rf_v2 = predict(modelo_rf_v2, rose_test, type='class')
# Confusion Matrix
```

```
confusionMatrix(rose_test$is_attributed, pred_rf_v2, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13503  1616
##          1   959 13919
##
##                Accuracy : 0.9142
##                  95% CI : (0.9109, 0.9173)
##     No Information Rate : 0.5179
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8284
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8960
##             Specificity : 0.9337
##          Pos Pred Value : 0.9355
##          Neg Pred Value : 0.8931
##              Prevalence : 0.5179
##          Detection Rate : 0.4640
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9148
##
##        'Positive' Class : 1
##
```

```
# Curva roc para o modelo_rf_v2
roc.curve(rose_test$is_attributed, pred_rf_v2, plotit = T,
          col = "grey", add.roc = T)
```

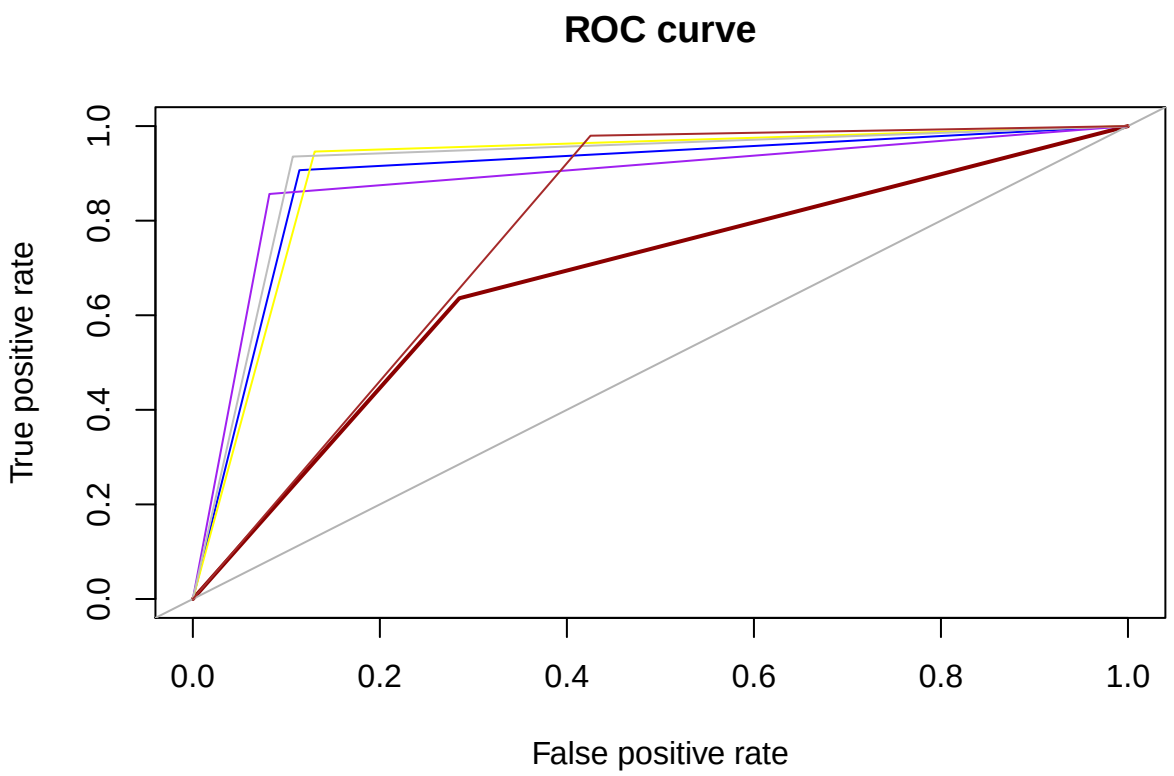
```
## Area under the curve (AUC): 0.914
```

```
# Criando o modelo com o algoritmo Naive Bayes
modelo_nb_v1 = naiveBayes(is_attributed ~ ., data=rose_train)
# Previsões nos dados de teste
pred_nb_v1 <- predict(modelo_nb_v1, newdata=rose_test)
# Confusion Matrix
confusionMatrix(rose_test$is_attributed, pred_nb_v1, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0  8690  6429
##          1   303 14575
##
##                Accuracy : 0.7756
##                  95% CI : (0.7708, 0.7803)
##     No Information Rate : 0.7002
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.5526
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6939
##             Specificity : 0.9663
##          Pos Pred Value : 0.9796
##          Neg Pred Value : 0.5748
##              Prevalence : 0.7002
##          Detection Rate : 0.4859
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.8301
##
##        'Positive' Class : 1
##
```

```r
# Curva roc para o modelo_nb_v1
roc.curve(rose_test$is_attributed, pred_nb_v1, plotit = T,
          col = "brown", add.roc = T)
```




```
## Area under the curve (AUC): 0.777
```

```r
# O melhor desempenho que tivemos foi do modelo_rf_v2, com random forest, vamos agora simular esse mode

# Lendo o arquivo test_p1 e fazendo a previsão com o agloritmo random forest
test_kaggle_p1 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
# transformando o arquivo para a predição
test_kaggle_p1 <- transform_test(test_kaggle_p1)
# realizando a predição
pred_rf_v2_p1 <- predict(modelo_rf_v2, test_kaggle_p1, type="response")
# criando um data frame com resposta e predição
```

```
prediction2_p1 <- data.frame('click_id' = test_kaggle_p1$click_id,
                             'is_attributed' = pred_rf_v2_p1)
# Visualizando o data frame
head(prediction2_p1)
```

```
##   click_id is_attributed
## 1        0             0
## 2        1             0
## 3        2             0
## 4        3             0
## 5        4             0
## 6        5             0
```

```
# Excluindo o arquivo de teste
rm(test_kaggle_p1)

# Lendo o arquivo test_p2 e fazendo a previsão com o agloritmo random forest
test_kaggle_p2 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
# transformando o arquivo para a predição
test_kaggle_p2 <- transform_test(test_kaggle_p2)
# realizando a predição
pred_rf_v2_p2 <- predict(modelo_rf_v2, test_kaggle_p2, type="response")
# criando um data frame com resposta e predição
prediction2_p2 <- data.frame('click_id' = test_kaggle_p2$click_id,
                             'is_attributed' = pred_rf_v2_p2)
# Visualizando o data frame
head(prediction2_p2)
```

```
##   click_id is_attributed
## 1  4697617             0
## 2  4697618             0
## 3  4697619             0
## 4  4697620             0
## 5  4697621             0
## 6  4697622             0
```

```
# Excluindo o arquivo de teste
rm(test_kaggle_p2)

# Lendo o arquivo test_p3 e fazendo a previsão com o agloritmo random forest
test_kaggle_p3 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
# transformando o arquivo para a predição
test_kaggle_p3 <- transform_test(test_kaggle_p3)
# realizando a predição
pred_rf_v2_p3 <- predict(modelo_rf_v2, test_kaggle_p3, type="response")
# criando um data frame com resposta e predição
prediction2_p3 <- data.frame('click_id' = test_kaggle_p3$click_id,
                             'is_attributed' = pred_rf_v2_p3)
# Visualizando o data frame
head(prediction2_p3)
```

```
##   click_id is_attributed
## 1  9395233             0
## 2  9395235             0
## 3  9395236             0
```

```
## 4  9395237             0
## 5  9395238             0
## 6  9395239             0
```

```
# Excluindo o arquivo de teste
rm(test_kaggle_p3)

# Lendo o arquivo test_p4 e fazendo a previsão com o agloritmo random forest
test_kaggle_p4 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
# transformando o arquivo para a predição
test_kaggle_p4 <- transform_test(test_kaggle_p4)
# realizando a predição
pred_rf_v2_p4 <- predict(modelo_rf_v2, test_kaggle_p4, type="response")
# criando um data frame com resposta e predição
prediction2_p4 <- data.frame('click_id' = test_kaggle_p4$click_id,
                             'is_attributed' = pred_rf_v2_p4)
# Visualizando o data frame
head(prediction2_p4)
```

```
##   click_id is_attributed
## 1 14092851             0
## 2 14092852             0
## 3 14092853             0
## 4 14092854             0
## 5 14092855             0
## 6 14092856             0
```

```
# Excluindo o arquivo de teste
rm(test_kaggle_p4)

# Juntando as 4 predições
prediction2 <- rbind(prediction2_p1, prediction2_p2,
                     prediction2_p3, prediction2_p4)

# Escrevendo o data set em um arquivo csv.
#fwrite(prediction2, "prediction2.csv", row.names = F, sep = ",")

# Excluindo alguns arquivos para limpara a memória do R.
rm(prediction2_p1, prediction2_p2, prediction2_p3, prediction2_p4, prediction2)
rm(pred_rf_v2_p1, pred_rf_v2_p2, pred_rf_v2_p3, pred_rf_v2_p4)
rm(model_glm_v2, modelo_tree_v2, modelo_svm_v1, modelo_rf_v1, modelo_rf_v2, modelo_nb_v1)
rm(pred_glm_v2, pred_tree_v2, pred_svm_v1, pred_rf_v1, pred_rf_v2, pred_nb_v1)
rm(train, teste, rose_train, rose_test)
```

```
## Warning in rm(train, teste, rose_train, rose_test): object 'teste' not found
```

```
# Para essa predição o score alcançado foi de 0,689, mas ainda não estou contente com o
# resultado, vamos tentar fazer mais algumas alterações no data set para ver se o algorítmo
# melhora
```

Terceiro Modelo Preditivo

Nosso segundo modelo preditivo teve uma ótima melhora em relação ao primeiro, mas ainda não é satisfatório. Nesta tereceira previsão vamos criar 4 novas colunas, count_apps, count_device, count_os e count_channel, que semelhantes ao ip contaram quantos tipos de cada cada variável existem. Assim o algoritmo conseguirá ver como as variáveis estão distribuídas numéricamente e assim ser mais preciso na previsão. Faremos a

previsão de duas formas antes de utilizar o arquivo teste. Na primeira usaremos os dados sem normalização e em seguida normalizaremos os dados para fazer a modelagem. O modelo que tiver o melhor desempenho será usado na previsão.

```
# Função para criar as novas variáveis.
num_vars <- function(df, col){
  df[[col]] <- as.character(df[[col]])
  count_col <- as.numeric(ave(df[[col]], df[[col]], FUN = length))
  return(count_col)
}

# Criando as novas variáveis
df$count_app <- num_vars(df, 'app')
df$count_device <- num_vars(df, 'device')
df$count_os <- num_vars(df, 'os')
df$count_channel <- num_vars(df, 'channel')

# Criando as variáveis train e test
train <- df[trainIndex,]
test <- df[-trainIndex,]

# Balanceando os dados
# Feature selection nos dados de treino e teste
train <- train %>% select(count_app, count_device, count_os, count_channel,
                          count_ip, range_hour, is_attributed)
test <- test %>% select(count_app, count_device, count_os, count_channel,
                        count_ip, range_hour, is_attributed)
# ROSE nos dados de treino
rose_train <- ROSE(is_attributed ~ ., data = train, seed = 1)$data
prop.table(table(rose_train$is_attributed))
```

```
##
##         0         1
## 0.4988142 0.5011858
```

```
# ROSE nos dados de teste
rose_test <- ROSE(is_attributed ~ ., data = test, seed = 1)$data
prop.table(table(rose_test$is_attributed))
```

```
##
##         0         1
## 0.5040171 0.4959829
```

```
# Treinando o modelo com o algoritmo de regressão logística
model_glm_v3 <- glm(is_attributed ~ ., data = rose_train, family = "binomial")
# Verificando alguns resultados do modelo treinado
summary(model_glm_v3)
```

```
##
## Call:
## glm(formula = is_attributed ~ ., family = "binomial", data = rose_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0243  -0.4785   0.1537   0.5575   4.3528
##
```

```
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          3.329e+00  3.786e-02  87.932   <2e-16 ***
## count_app           -2.325e-04  2.417e-06 -96.203   <2e-16 ***
## count_device        -1.354e-05  3.179e-07 -42.604   <2e-16 ***
## count_os            -1.106e-05  1.034e-06 -10.700   <2e-16 ***
## count_channel       -6.404e-04  1.017e-05 -63.000   <2e-16 ***
## count_ip            -1.780e-02  5.172e-04 -34.416   <2e-16 ***
## range_hour4 as 8     1.468e-02  3.289e-02   0.446   0.6554
## range_hour8 as 12   -3.480e-01  3.335e-02 -10.436   <2e-16 ***
## range_hour12 as 16  -4.414e-01  3.363e-02 -13.127   <2e-16 ***
## range_hour16 as 20  -1.575e+00  7.056e-02 -22.321   <2e-16 ***
## range_hour16 as 24   7.741e-02  4.392e-02   1.763   0.0779 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 97035  on 69995  degrees of freedom
## Residual deviance: 51429  on 69985  degrees of freedom
## AIC: 51451
##
## Number of Fisher Scoring iterations: 6
```

```
# Realizando a predição com o modelo treinado
pred_glm_v3 <- predict(model_glm_v3, rose_test, type="response")
# Arredondando para 0 ou 1
pred_glm_v3 <- round(pred_glm_v3)
#Confusion Matrix da predição.
confusionMatrix(table(data = pred_glm_v3, reference = rose_test$is_attributed),
                positive = '1')
```

```
## Confusion Matrix and Statistics
##
##     reference
## data     0     1
##    0 12726  1481
##    1  2393 13397
##
##                Accuracy : 0.8709
##                  95% CI : (0.867, 0.8746)
##     No Information Rate : 0.504
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7418
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9005
##             Specificity : 0.8417
##          Pos Pred Value : 0.8484
##          Neg Pred Value : 0.8958
##              Prevalence : 0.4960
##          Detection Rate : 0.4466
##    Detection Prevalence : 0.5264
```

```
##       Balanced Accuracy : 0.8711
##
##        'Positive' Class : 1
##
```

```
# Curva roc para o model_glm_v3
roc.curve(rose_test$is_attributed, pred_glm_v3, plotit = T,
          col = "darkred")
```

```
## Area under the curve (AUC): 0.871
```

```
# Criando o modelo com o algoritmo Árvore de Decissão
modelo_tree_v3 = C5.0(is_attributed ~ ., data = rose_train)
# Previsões nos dados de teste
pred_tree_v3 = predict(modelo_tree_v3, rose_test, type='class')
# Confusion Matrix
confusionMatrix(rose_test$is_attributed, pred_tree_v3, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14341   778
##          1   976 13902
##
##                Accuracy : 0.9415
##                  95% CI : (0.9388, 0.9442)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.883
##
##  Mcnemar's Test P-Value : 2.553e-06
##
##             Sensitivity : 0.9470
##             Specificity : 0.9363
##          Pos Pred Value : 0.9344
##          Neg Pred Value : 0.9485
##              Prevalence : 0.4894
##          Detection Rate : 0.4634
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9416
##
##        'Positive' Class : 1
##
```

```
# Previsão nos dados de teste
pred_tree_v3 <- as.numeric(pred_tree_v3)
# Curva roc para o modelo_tree_v3
roc.curve(rose_test$is_attributed, pred_tree_v3, plotit = T,
          col = "blue", add.roc = T)
```

```
## Area under the curve (AUC): 0.941
```

```
# Criando o modelo com o algoritmo SVM (Suport Vector Machine)
modelo_svm_v2 <- svm(is_attributed ~ ., data = rose_train,
                     type = 'C-classification', kernel = 'radial')
```

```r
# Previsão nos dados de teste
pred_svm_v2 = predict(modelo_svm_v2, rose_test)
# Confusion Matrix
confusionMatrix(rose_test$is_attributed, pred_svm_v2, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13935  1184
##          1  1311 13567
##
##                Accuracy : 0.9168
##                  95% CI : (0.9136, 0.9199)
##     No Information Rate : 0.5083
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8336
##
##  Mcnemar's Test P-Value : 0.01165
##
##             Sensitivity : 0.9197
##             Specificity : 0.9140
##          Pos Pred Value : 0.9119
##          Neg Pred Value : 0.9217
##              Prevalence : 0.4917
##          Detection Rate : 0.4523
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9169
##
##        'Positive' Class : 1
##
```

```r
# Curva roc para o modelo_svm_v2
roc.curve(rose_test$is_attributed, pred_svm_v2, plotit = T,
          col = "purple", add.roc = T)
```

```
## Area under the curve (AUC): 0.917
```

```r
# Criando o modelo com o algoritmo Random Forest
modelo_rf_v3 = rpart(is_attributed ~ ., data = rose_train,
                     control = rpart.control(cp = .0005))
# Previsões nos dados de teste
pred_rf_v3 = predict(modelo_rf_v3, rose_test, type='class')
# Confusion Matrix
confusionMatrix(rose_test$is_attributed, pred_rf_v3, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14242   877
##          1   646 14232
##
##                Accuracy : 0.9492
```

```
##                  95% CI : (0.9467, 0.9517)
##     No Information Rate : 0.5037
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8985
##
##  Mcnemar's Test P-Value : 3.78e-09
##
##             Sensitivity : 0.9420
##             Specificity : 0.9566
##          Pos Pred Value : 0.9566
##          Neg Pred Value : 0.9420
##              Prevalence : 0.5037
##          Detection Rate : 0.4744
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9493
##
##        'Positive' Class : 1
##
```

```
# Curva roc para o modelo_rf_v3
roc.curve(rose_test$is_attributed, pred_rf_v3, plotit = T,
          col = "yellow", add.roc = T)
```

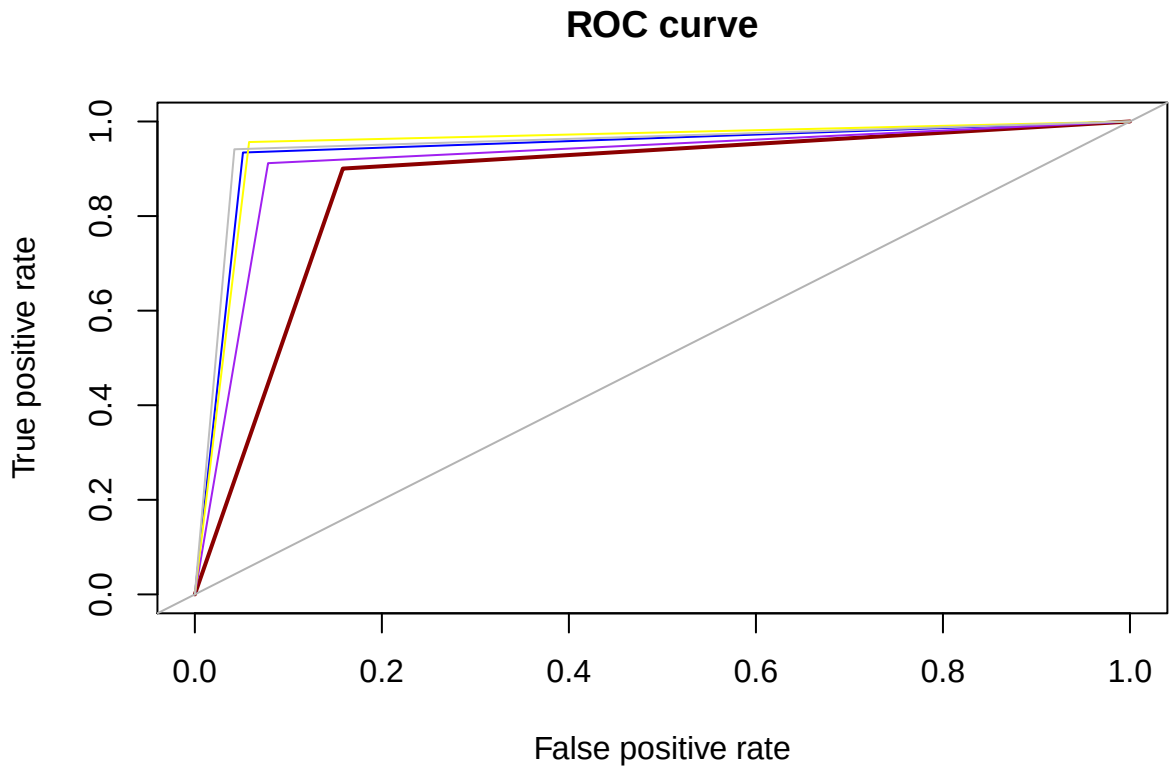
```
## Area under the curve (AUC): 0.949
```

```
# Criando o modelo com outro algoritmo Random Forest
modelo_rf_v4 = randomForest(formula = is_attributed ~ ., data = rose_train,
                            importance = TRUE)
# Previsões nos dados de teste
pred_rf_v4 = predict(modelo_rf_v4, rose_test, type='class')
# Confusion Matrix
confusionMatrix(rose_test$is_attributed, pred_rf_v4, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14480   639
##          1   874 14004
##
##                Accuracy : 0.9496
##                  95% CI : (0.947, 0.952)
##     No Information Rate : 0.5119
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8991
##
##  Mcnemar's Test P-Value : 1.79e-09
##
##             Sensitivity : 0.9564
##             Specificity : 0.9431
##          Pos Pred Value : 0.9413
##          Neg Pred Value : 0.9577
##              Prevalence : 0.4881
##          Detection Rate : 0.4668
```

```
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9497
##
##        'Positive' Class : 1
##
```

```r
# Curva roc para o modelo_rf_v4
roc.curve(rose_test$is_attributed, pred_rf_v4, plotit = T,
          col = "grey", add.roc = T)
```




```
## Area under the curve (AUC): 0.949
```

```r
# Esses valores estão ótimos, mas vamos normalizar os dados para ver se melhora a previsão.

# Copiando o data frame original
norm.df <- df

# Função para normalizar os dados
normdf <- function(df, col){
  df[[col]] <- scale(df[[col]])
  return(df)
}

# Normalizando os dados numéricos
norm.df <- normdf(norm.df, 'count_app')
norm.df <- normdf(norm.df, 'count_device')
norm.df <- normdf(norm.df, 'count_os')
norm.df <- normdf(norm.df, 'count_channel')
norm.df <- normdf(norm.df, 'count_ip')

# Criando as variáveis train e test
```

```r
norm.train <- norm.df[trainIndex,]
norm.test <- norm.df[-trainIndex,]

# Feature selection nos dados de treino e teste
norm.train <- norm.train %>% select(count_app, count_device, count_os,
                                    count_channel, count_ip, range_hour,
                                    is_attributed)
norm.test <- norm.test %>% select(count_app, count_device, count_os,
                                  count_channel, count_ip, range_hour,
                                  is_attributed)

# ROSE nos dados de treino
norm.rose_train <- ROSE(is_attributed ~ ., data = norm.train, seed = 1)$data
prop.table(table(norm.rose_train$is_attributed))
```

```
##
##         0         1
## 0.4988142 0.5011858
```

```r
# ROSE nos dados de teste
norm.rose_test <- ROSE(is_attributed ~ ., data = norm.test, seed = 1)$data
prop.table(table(norm.rose_test$is_attributed))
```

```
##
##         0         1
## 0.5040171 0.4959829
```

```r
# Treinando o modelo com o algoritmo de regressão logística
model_glm_v4 <- glm(is_attributed ~ ., data = norm.rose_train, family = "binomial")
# Verificando alguns resultados do modelo treinado
summary(model_glm_v4)
```

```
##
## Call:
## glm(formula = is_attributed ~ ., family = "binomial", data = norm.rose_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0243  -0.4785   0.1537   0.5575   4.3528
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.202566   0.031664 -69.560   <2e-16 ***
## count_app          -1.353453   0.014069 -96.203   <2e-16 ***
## count_device       -0.284542   0.006679 -42.604   <2e-16 ***
## count_os           -0.112375   0.010502 -10.700   <2e-16 ***
## count_channel      -1.368550   0.021723 -63.000   <2e-16 ***
## count_ip           -1.525234   0.044317 -34.416   <2e-16 ***
## range_hour4 as 8    0.014681   0.032892   0.446   0.6554
## range_hour8 as 12  -0.348038   0.033349 -10.436   <2e-16 ***
## range_hour12 as 16 -0.441400   0.033626 -13.127   <2e-16 ***
## range_hour16 as 20 -1.574987   0.070560 -22.321   <2e-16 ***
## range_hour16 as 24  0.077412   0.043915   1.763   0.0779 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 97035  on 69995  degrees of freedom
## Residual deviance: 51429  on 69985  degrees of freedom
## AIC: 51451
##
## Number of Fisher Scoring iterations: 6
```

```
# Realizando a predição com o modelo treinado
pred_glm_v4 <- predict(model_glm_v4, norm.rose_test, type="response")
# Arredondando para 0 ou 1
pred_glm_v4 <- round(pred_glm_v4)
#Confusion Matrix da predição.
confusionMatrix(table(data = pred_glm_v4, reference = norm.rose_test$is_attributed),
                positive = '1')
```

```
## Confusion Matrix and Statistics
##
##     reference
## data     0     1
##    0 12726  1481
##    1  2393 13397
##
##                Accuracy : 0.8709
##                  95% CI : (0.867, 0.8746)
##     No Information Rate : 0.504
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7418
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9005
##             Specificity : 0.8417
##          Pos Pred Value : 0.8484
##          Neg Pred Value : 0.8958
##              Prevalence : 0.4960
##          Detection Rate : 0.4466
##    Detection Prevalence : 0.5264
##       Balanced Accuracy : 0.8711
##
##        'Positive' Class : 1
##
```

```
# Curva roc para o model_glm_v4
roc.curve(norm.rose_test$is_attributed, pred_glm_v4, plotit = T,
          col = "darkred")
```

```
## Area under the curve (AUC): 0.871
```

```
# Criando o modelo com o algoritmo Árvore de Decissão
modelo_tree_v4 = C5.0(is_attributed ~ ., data = norm.rose_train)
# Previsão nos dados de teste
pred_tree_v4 = predict(modelo_tree_v4, norm.rose_test, type='class')
# Confusion Matrix
```

```r
confusionMatrix(norm.rose_test$is_attributed, pred_tree_v4, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14341   778
##          1   976 13902
##
##                Accuracy : 0.9415
##                  95% CI : (0.9388, 0.9442)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.883
##
##  Mcnemar's Test P-Value : 2.553e-06
##
##             Sensitivity : 0.9470
##             Specificity : 0.9363
##          Pos Pred Value : 0.9344
##          Neg Pred Value : 0.9485
##              Prevalence : 0.4894
##          Detection Rate : 0.4634
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9416
##
##        'Positive' Class : 1
##
```

```r
# Convertendo para numérico
pred_tree_v4 <- as.numeric(pred_tree_v4)
# Curva roc para o modelo_tree_v4
roc.curve(norm.rose_test$is_attributed, pred_tree_v4, plotit = T,
          col = "blue", add.roc = T)
```

```
## Area under the curve (AUC): 0.941
```

```r
# Criando o modelo com o algoritmo SVM (Suport Vector Machine)
modelo_svm_v3 <- svm(is_attributed ~ ., data = norm.rose_train,
                     type = 'C-classification', kernel = 'radial')
# Previsão nos dados de teste
pred_svm_v3 = predict(modelo_svm_v3, norm.rose_test)
# Confusion Matrix
confusionMatrix(norm.rose_test$is_attributed, pred_svm_v3, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 13935  1184
##          1  1311 13567
##
##                Accuracy : 0.9168
##                  95% CI : (0.9136, 0.9199)
```

```
##     No Information Rate : 0.5083
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8336
##
##  Mcnemar's Test P-Value : 0.01165
##
##             Sensitivity : 0.9197
##             Specificity : 0.9140
##          Pos Pred Value : 0.9119
##          Neg Pred Value : 0.9217
##              Prevalence : 0.4917
##          Detection Rate : 0.4523
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9169
##
##        'Positive' Class : 1
##
```

```
# Curva roc para o modelo_svm_v3
roc.curve(norm.rose_test$is_attributed, pred_svm_v3, plotit = T,
          col = "purple", add.roc = T)
```

```
## Area under the curve (AUC): 0.917
```

```
# Criando o modelo com o algoritmo Random Forest
modelo_rf_v5 = rpart(is_attributed ~ ., data = norm.rose_train,
                     control = rpart.control(cp = .0005))
# Previsão nos dados de teste
pred_rf_v5 = predict(modelo_rf_v5, norm.rose_test, type='class')
# Confusion Matrix
confusionMatrix(norm.rose_test$is_attributed, pred_rf_v5, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14242   877
##          1   646 14232
##
##                Accuracy : 0.9492
##                  95% CI : (0.9467, 0.9517)
##     No Information Rate : 0.5037
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8985
##
##  Mcnemar's Test P-Value : 3.78e-09
##
##             Sensitivity : 0.9420
##             Specificity : 0.9566
##          Pos Pred Value : 0.9566
##          Neg Pred Value : 0.9420
##              Prevalence : 0.5037
##          Detection Rate : 0.4744
##    Detection Prevalence : 0.4960
```

```
##       Balanced Accuracy : 0.9493
##
##        'Positive' Class : 1
##
```

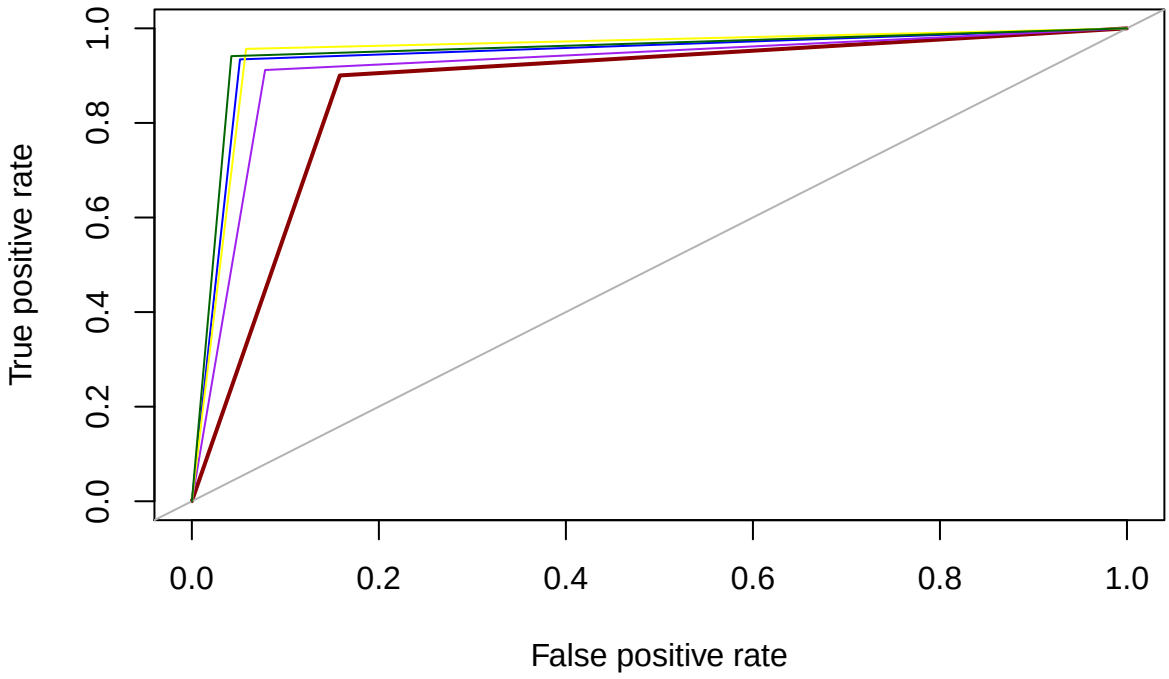
```
# Curva roc para o modelo_rf_v5
roc.curve(norm.rose_test$is_attributed, pred_rf_v5, plotit = T,
          col = "yellow", add.roc = T)
```

```
## Area under the curve (AUC): 0.949
```

```
# Criando o modelo com outro algoritmo Random Forest
modelo_rf_v6 = randomForest(formula = is_attributed ~ ., data = norm.rose_train,
                            importance = TRUE)
# Previsão nos dados de teste
pred_rf_v6 = predict(modelo_rf_v6, norm.rose_test, type='class')
# Confusion Matrix
confusionMatrix(norm.rose_test$is_attributed, pred_rf_v6, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14480   639
##          1   874 14004
##
##                Accuracy : 0.9496
##                  95% CI : (0.947, 0.952)
##     No Information Rate : 0.5119
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8991
##
##  Mcnemar's Test P-Value : 1.79e-09
##
##             Sensitivity : 0.9564
##             Specificity : 0.9431
##          Pos Pred Value : 0.9413
##          Neg Pred Value : 0.9577
##              Prevalence : 0.4881
##          Detection Rate : 0.4668
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9497
##
##        'Positive' Class : 1
##
```

```
# Curva roc para o modelo_v1
roc.curve(norm.rose_test$is_attributed, pred_rf_v6, plotit = T,
          col = "darkgreen", add.roc = T)
```




```
## Area under the curve (AUC): 0.949
```

```
# Como os resultados foram piores do que sem a normalização, vou utilizar o modelo_rf_v4 (sem a normali

#Limpando alguns dados que não serão mais usados
rm(model_glm_v3, modelo_tree_v3, modelo_svm_v2, modelo_rf_v3, model_glm_v4,
   modelo_tree_v4, modelo_svm_v3, modelo_rf_v5, modelo_rf_v6)
rm(pred_glm_v3, pred_tree_v3, pred_svm_v2, pred_rf_v3, pred_glm_v4, pred_tree_v4,
   pred_svm_v3, pred_rf_v5, pred_rf_v6)
rm(norm.df, norm.train, norm.test, norm.rose_train, norm.rose_test)

# Antes vamos criar uma função que conta a quantidade de valores em cada variável que modificamos com c

# Função para que retorna um data set valores e quantidade de cada valor
count_vars <- function(df, var){
  filtra_var <- df %>%
    group_by_(var) %>%
    summarise(count = n())
  return(filtra_var)
}

# Criando o novo data set.
dfapp <- count_vars(df, 'app')
```

```
## Warning: group_by_() is deprecated.
## Please use group_by() instead
##
## The 'programming' vignette or the tidyeval book can help you
## to program with group_by() : https://tidyeval.tidyverse.org
## This warning is displayed once per session.
```

```r
dfdevice <- count_vars(df, 'device')
dfos <- count_vars(df, 'os')
dfchannel <- count_vars(df, 'channel')

# Função para calcular a moda
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Verificando quais quantidades mais aparecem em cada variável.
getmode(dfapp$count)
```

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

```r
getmode(dfdevice$count)
```

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

```r
getmode(dfos$count)
```

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

```r
getmode(dfchannel$count)
```

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

```r
#Como podemos comprovar o valor 1 é o valor mais frequente nas variáveis counts

# Lendo o arquivo parte 1
test_kaggle_p1 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
test_kaggle_p1 <- transform_test(test_kaggle_p1)
# Criando a coluna count_app
test_kaggle_p1$count_app <- dfapp$count[match(test_kaggle_p1$app, dfapp$app)]
# Renumerando os valores NA`s para 1
test_kaggle_p1[(is.na(test_kaggle_p1$count_app) == TRUE), 'count_app'] = 1
# Criando a coluna count_device
test_kaggle_p1$count_device <- dfdevice$count[match(test_kaggle_p1$device, dfdevice$device)]
# Renumerando os valores NA`s para 1
test_kaggle_p1[(is.na(test_kaggle_p1$count_device) == TRUE), 'count_device'] = 1
# Criando a coluna count_os
test_kaggle_p1$count_os <- dfos$count[match(test_kaggle_p1$os,dfos$os)]
# Renumerando os valores NA`s para 1
test_kaggle_p1[(is.na(test_kaggle_p1$count_os) == TRUE), 'count_os'] = 1
# Criando a coluna count_app
test_kaggle_p1$count_channel <- dfchannel$count[match(test_kaggle_p1$channel,dfchannel$channel)]
# Renumerando os valores NA`s para 1
test_kaggle_p1[(is.na(test_kaggle_p1$count_channel) == TRUE), 'count_channel'] = 1
# Criando um data set com o real e o previsto
test_kaggle_p1 <- test_kaggle_p1[, c('click_id', 'count_app', 'count_device', 'count_os',
                                     'count_channel', 'count_ip', 'range_hour')]
# Realizando a previsão
pred_rf_v3_p1 <- predict(modelo_rf_v4, test_kaggle_p1, type="response")
# Criando um data set com o real e o previsto
prediction3_p1 <- data.frame('click_id' = test_kaggle_p1$click_id,
                             'is_attributed' = pred_rf_v3_p1)
```

```r
# Visualizando os primeiros dados
head(prediction3_p1)
```

```
##   click_id is_attributed
## 1        0             0
## 2        1             0
## 3        2             0
## 4        3             0
## 5        4             0
## 6        5             0
```

```r
# Excluindo o data set de teste
rm(test_kaggle_p1)


# Lendo o arquivo parte 2
test_kaggle_p2 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
test_kaggle_p2 <- transform_test(test_kaggle_p2)
# Criando a coluna count_app
test_kaggle_p2$count_app <- dfapp$count[match(test_kaggle_p2$app, dfapp$app)]
# Renumerando os valores NA`s para 1
test_kaggle_p2[(is.na(test_kaggle_p2$count_app) == TRUE), 'count_app'] = 1
# Criando a coluna count_device
test_kaggle_p2$count_device <- dfdevice$count[match(test_kaggle_p2$device, dfdevice$device)]
# Renumerando os valores NA`s para 1
test_kaggle_p2[(is.na(test_kaggle_p2$count_device) == TRUE), 'count_device'] = 1
# Criando a coluna count_os
test_kaggle_p2$count_os <- dfos$count[match(test_kaggle_p2$os, dfos$os)]
# Renumerando os valores NA`s para 1
test_kaggle_p2[(is.na(test_kaggle_p2$count_os) == TRUE), 'count_os'] = 1
# Criando a coluna count_app
test_kaggle_p2$count_channel <- dfchannel$count[match(test_kaggle_p2$channel, dfchannel$channel)]
# Renumerando os valores NA`s para 1
test_kaggle_p2[(is.na(test_kaggle_p2$count_channel) == TRUE), 'count_channel'] = 1
test_kaggle_p2 <- test_kaggle_p2[, c('click_id', 'count_app', 'count_device', 'count_os',
                                     'count_channel', 'count_ip', 'range_hour')]
# Realizando a previsão
pred_rf_v3_p2 <- predict(modelo_rf_v4, test_kaggle_p2, type="response")
# Criando um data set com o real e o previsto
prediction3_p2 <- data.frame('click_id' = test_kaggle_p2$click_id,
                             'is_attributed' = pred_rf_v3_p2)
# Visualizando os primeiros dados
head(prediction3_p2)
```

```
##   click_id is_attributed
## 1  4697617             0
## 2  4697618             0
## 3  4697619             0
## 4  4697620             0
## 5  4697621             0
## 6  4697622             0
```

```r
# Excluindo o data set de teste
rm(test_kaggle_p2)
```

```
# Lendo o arquivo parte 3
test_kaggle_p3 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
test_kaggle_p3 <- transform_test(test_kaggle_p3)
# Criando a coluna count_app
test_kaggle_p3$count_app <- dfapp$count[match(test_kaggle_p3$app, dfapp$app)]
# Renumerando os valores NA`s para 1
test_kaggle_p3[(is.na(test_kaggle_p3$count_app) == TRUE), 'count_app'] = 1
# Criando a coluna count_device
test_kaggle_p3$count_device <- dfdevice$count[match(test_kaggle_p3$device, dfdevice$device)]
# Renumerando os valores NA`s para 1
test_kaggle_p3[(is.na(test_kaggle_p3$count_device) == TRUE), 'count_device'] = 1
# Criando a coluna count_os
test_kaggle_p3$count_os <- dfos$count[match(test_kaggle_p3$os, dfos$os)]
# Renumerando os valores NA`s para 1
test_kaggle_p3[(is.na(test_kaggle_p3$count_os) == TRUE), 'count_os'] = 1
# Criando a coluna count_app
test_kaggle_p3$count_channel <- dfchannel$count[match(test_kaggle_p3$channel, dfchannel$channel)]
# Renumerando os valores NA`s para 1
test_kaggle_p3[(is.na(test_kaggle_p3$count_channel) == TRUE), 'count_channel'] = 1
test_kaggle_p3 <- test_kaggle_p3[, c('click_id', 'count_app', 'count_device', 'count_os',
                                     'count_channel', 'count_ip', 'range_hour')]
# Realizando a previsão
pred_rf_v3_p3 <- predict(modelo_rf_v4, test_kaggle_p3, type="response")
# Criando um data set com o real e o previsto
prediction3_p3 <- data.frame('click_id' = test_kaggle_p3$click_id,
                             'is_attributed' = pred_rf_v3_p3)
# Visualizando os primeiros dados
head(prediction3_p3)
```

```
##   click_id is_attributed
## 1  9395233             0
## 2  9395235             0
## 3  9395236             0
## 4  9395237             0
## 5  9395238             0
## 6  9395239             0
```

```
# Excluindo o data set de teste
rm(test_kaggle_p3)


# Lendo o arquivo parte 4
test_kaggle_p4 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
test_kaggle_p4 <- transform_test(test_kaggle_p4)
# Criando a coluna count_app
test_kaggle_p4$count_app <- dfapp$count[match(test_kaggle_p4$app, dfapp$app)]
# Renumerando os valores NA`s para 1
test_kaggle_p4[(is.na(test_kaggle_p4$count_app) == TRUE), 'count_app'] = 1
# Criando a coluna count_device
test_kaggle_p4$count_device <- dfdevice$count[match(test_kaggle_p4$device, dfdevice$device)]
# Renumerando os valores NA`s para 1
test_kaggle_p4[(is.na(test_kaggle_p4$count_device) == TRUE), 'count_device'] = 1
# Criando a coluna count_os
test_kaggle_p4$count_os <- dfos$count[match(test_kaggle_p4$os, dfos$os)]
```

```
# Renumerando os valores NA`s para 1
test_kaggle_p4[(is.na(test_kaggle_p4$count_os) == TRUE), 'count_os'] = 1
# Criando a coluna count_app
test_kaggle_p4$count_channel <- dfchannel$count[match(test_kaggle_p4$channel, dfchannel$channel)]
# Renumerando os valores NA`s para 1
test_kaggle_p4[(is.na(test_kaggle_p4$count_channel) == TRUE), 'count_channel'] = 1
test_kaggle_p4 <- test_kaggle_p4[, c('click_id', 'count_app', 'count_device', 'count_os',
                                     'count_channel', 'count_ip', 'range_hour')]
# Realizando a previsão
pred_rf_v3_p4 <- predict(modelo_rf_v4, test_kaggle_p4, type="response")
# Criando um data set com o real e o previsto
prediction3_p4 <- data.frame('click_id' = test_kaggle_p4$click_id,
                             'is_attributed' = pred_rf_v3_p4)
# Visualizando os primeiros dados
head(prediction3_p4)
```

```
##   click_id is_attributed
## 1 14092851             0
## 2 14092852             0
## 3 14092853             0
## 4 14092854             0
## 5 14092855             0
## 6 14092856             0
```

```
# Excluindo o data set de teste
rm(test_kaggle_p4)

# Juntando as 4 predições
prediction3 <- rbind(prediction3_p1, prediction3_p2, prediction3_p3, prediction3_p4)
# Escrevendo o data set em um arquivo csv.
#fwrite(prediction3, "prediction3.csv", row.names = F, sep = ",")

# Excluindo algumas variáveis que não serão mais utilizadas
rm(prediction3_p1, prediction3_p2, prediction3_p3, prediction3_p4)
rm(pred_rf_v3_p1, pred_rf_v3_p2, pred_rf_v3_p3, pred_rf_v3_p4)
rm(modelo_rf_v4, pred_rf_v4)
rm(prediction3)

# O score dessa previsão ficou em 0.72 no kaggle, uma pequena melhora em relação ao algoritmo anterior.
```

Quarto Modelo Preditivo

Neste quarto modelo preditivo vamos utilizar, além das colunas counts as variáveis relacionadas a elas, menos a ip, pois não nos da muita informação. Quanto as outras variáveis o algorítmo saberá quais têm a mesma quantidade, mas que tem mais downloads.

```
# Criando as variáveis train e test
train <- df[trainIndex,]
test <- df[-trainIndex,]

# Balanceando os dados
# Feature selection nos dados de treino e teste
train <- train %>% select(app, device, os, channel, count_app, count_device, count_os,
                          count_channel, count_ip, range_hour, is_attributed)
test <- test %>% select(app, device, os, channel, count_app, count_device, count_os,
```

```
                        count_channel, count_ip, range_hour, is_attributed)
# ROSE nos dados de treino
rose_train <- ROSE(is_attributed ~ ., data = train, seed = 1)$data
prop.table(table(rose_train$is_attributed))
```

```
##
##         0         1
## 0.4988142 0.5011858
```

```
# ROSE nos dados de teste
rose_test <- ROSE(is_attributed ~ ., data = test, seed = 1)$data
prop.table(table(rose_test$is_attributed))
```

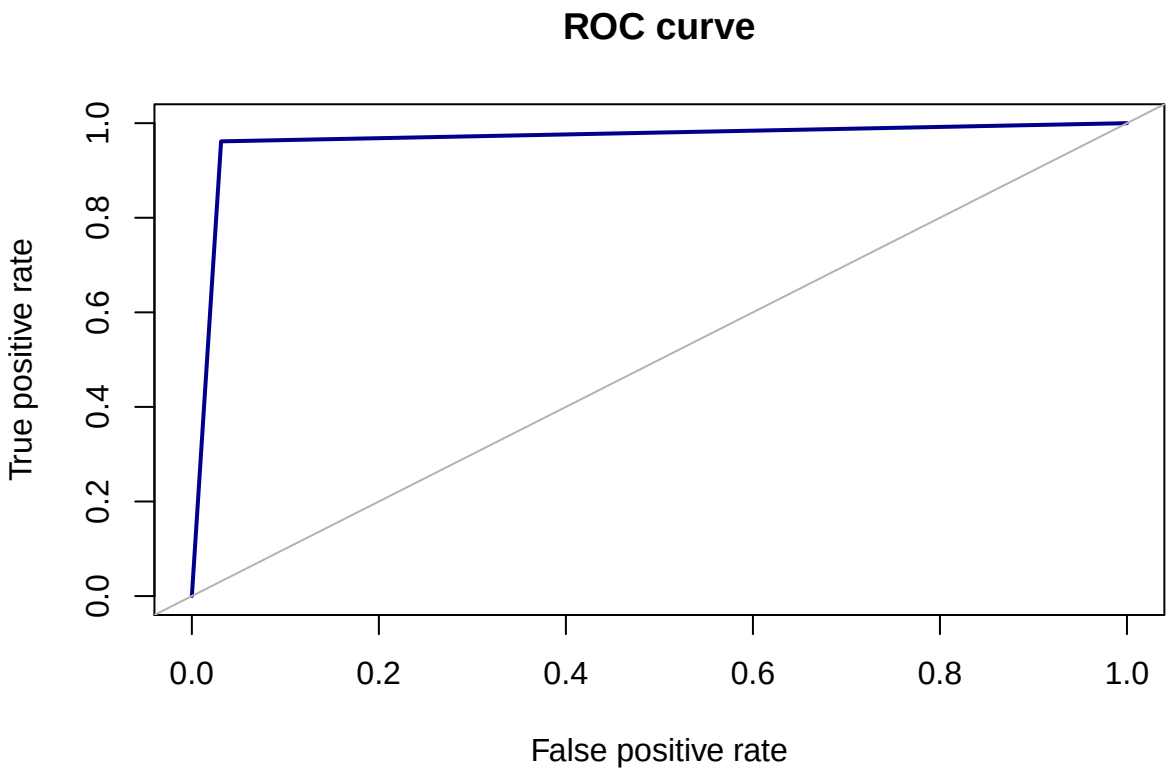
```
##
##         0         1
## 0.5040171 0.4959829
```

```
# Vamos treinar o modelo apenas com o algoritmo random forest, pois foi o que teve os melhores resultad
# Criando o modelo com outro algoritmo Random Forest
modelo_rf_v7 = randomForest(formula = is_attributed ~ ., data = rose_train,
                            importance = TRUE)
# Previsão nos dados de teste
pred_rf_v7 = predict(modelo_rf_v7, rose_test, type='class')
# Confusion Matrix
confusionMatrix(rose_test$is_attributed, pred_rf_v7, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14646   473
##          1   572 14306
##
##                Accuracy : 0.9652
##                  95% CI : (0.963, 0.9672)
##     No Information Rate : 0.5073
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9303
##
##  Mcnemar's Test P-Value : 0.002433
##
##             Sensitivity : 0.9680
##             Specificity : 0.9624
##          Pos Pred Value : 0.9616
##          Neg Pred Value : 0.9687
##              Prevalence : 0.4927
##          Detection Rate : 0.4769
##    Detection Prevalence : 0.4960
##       Balanced Accuracy : 0.9652
##
##        'Positive' Class : 1
##
```

```
# Curva roc para o modelo_v1
roc.curve(rose_test$is_attributed, pred_rf_v7, plotit = T,
```

```
          col = "darkblue")
```




```
## Area under the curve (AUC): 0.965
```

```
# Lendo o arquivo parte 1
test_kaggle_p1 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
test_kaggle_p1 <- transform_test(test_kaggle_p1)
# Criando a coluna count_app
test_kaggle_p1$count_app <- dfapp$count[match(test_kaggle_p1$app, dfapp$app)]
# Renumerando os valores NA`s para 1
test_kaggle_p1[(is.na(test_kaggle_p1$count_app) == TRUE), 'count_app'] = 1
# Criando a coluna count_device
test_kaggle_p1$count_device <- dfdevice$count[match(test_kaggle_p1$device, dfdevice$device)]
# Renumerando os valores NA`s para 1
test_kaggle_p1[(is.na(test_kaggle_p1$count_device) == TRUE), 'count_device'] = 1
# Criando a coluna count_os
test_kaggle_p1$count_os <- dfos$count[match(test_kaggle_p1$os,dfos$os)]
# Renumerando os valores NA`s para 1
test_kaggle_p1[(is.na(test_kaggle_p1$count_os) == TRUE), 'count_os'] = 1
# Criando a coluna count_app
test_kaggle_p1$count_channel <- dfchannel$count[match(test_kaggle_p1$channel,dfchannel$channel)]
# Renumerando os valores NA`s para 1
test_kaggle_p1[(is.na(test_kaggle_p1$count_channel) == TRUE), 'count_channel'] = 1
# Criando um data set com o real e o previsto
test_kaggle_p1 <- test_kaggle_p1[, c('app', 'device', 'os', 'channel', 'click_id', 'count_app',
                                     'count_device', 'count_os', 'count_channel', 'count_ip', 'range_hou
# Realizando a previsão
pred_rf_v4_p1 <- predict(modelo_rf_v7, test_kaggle_p1, type="response")
# Criando um data set com o real e o previsto
prediction4_p1 <- data.frame('click_id' = test_kaggle_p1$click_id,
```

```
                              'is_attributed' = pred_rf_v4_p1)
# Visualizando os primeiros dados
head(prediction4_p1)
```

```
##   click_id is_attributed
## 1        0             0
## 2        1             0
## 3        2             0
## 4        3             0
## 5        4             0
## 6        5             0
```

```
# Excluindo o data set de teste
rm(test_kaggle_p1)


# Lendo o arquivo parte 2
test_kaggle_p2 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
test_kaggle_p2 <- transform_test(test_kaggle_p2)
# Criando a coluna count_app
test_kaggle_p2$count_app <- dfapp$count[match(test_kaggle_p2$app, dfapp$app)]
# Renumerando os valores NA`s para 1
test_kaggle_p2[(is.na(test_kaggle_p2$count_app) == TRUE), 'count_app'] = 1
# Criando a coluna count_device
test_kaggle_p2$count_device <- dfdevice$count[match(test_kaggle_p2$device, dfdevice$device)]
# Renumerando os valores NA`s para 1
test_kaggle_p2[(is.na(test_kaggle_p2$count_device) == TRUE), 'count_device'] = 1
# Criando a coluna count_os
test_kaggle_p2$count_os <- dfos$count[match(test_kaggle_p2$os, dfos$os)]
# Renumerando os valores NA`s para 1
test_kaggle_p2[(is.na(test_kaggle_p2$count_os) == TRUE), 'count_os'] = 1
# Criando a coluna count_app
test_kaggle_p2$count_channel <- dfchannel$count[match(test_kaggle_p2$channel, dfchannel$channel)]
# Renumerando os valores NA`s para 1
test_kaggle_p2[(is.na(test_kaggle_p2$count_channel) == TRUE), 'count_channel'] = 1
test_kaggle_p2 <- test_kaggle_p2[, c('app', 'device', 'os', 'channel', 'click_id', 'count_app',
                                     'count_device', 'count_os', 'count_channel', 'count_ip', 'range_hou
# Realizando a previsão
pred_rf_v4_p2 <- predict(modelo_rf_v7, test_kaggle_p2, type="response")
# Criando um data set com o real e o previsto
prediction4_p2 <- data.frame('click_id' = test_kaggle_p2$click_id,
                             'is_attributed' = pred_rf_v4_p2)
# Visualizando os primeiros dados
head(prediction4_p2)
```

```
##   click_id is_attributed
## 1  4697617             0
## 2  4697618             0
## 3  4697619             0
## 4  4697620             0
## 5  4697621             0
## 6  4697622             0
```

```
# Excluindo o data set de teste
rm(test_kaggle_p2)
```

```
# Lendo o arquivo parte 3
test_kaggle_p3 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
test_kaggle_p3 <- transform_test(test_kaggle_p3)
# Criando a coluna count_app
test_kaggle_p3$count_app <- dfapp$count[match(test_kaggle_p3$app, dfapp$app)]
# Renumerando os valores NA`s para 1
test_kaggle_p3[(is.na(test_kaggle_p3$count_app) == TRUE), 'count_app'] = 1
# Criando a coluna count_device
test_kaggle_p3$count_device <- dfdevice$count[match(test_kaggle_p3$device, dfdevice$device)]
# Renumerando os valores NA`s para 1
test_kaggle_p3[(is.na(test_kaggle_p3$count_device) == TRUE), 'count_device'] = 1
# Criando a coluna count_os
test_kaggle_p3$count_os <- dfos$count[match(test_kaggle_p3$os, dfos$os)]
# Renumerando os valores NA`s para 1
test_kaggle_p3[(is.na(test_kaggle_p3$count_os) == TRUE), 'count_os'] = 1
# Criando a coluna count_app
test_kaggle_p3$count_channel <- dfchannel$count[match(test_kaggle_p3$channel, dfchannel$channel)]
# Renumerando os valores NA`s para 1
test_kaggle_p3[(is.na(test_kaggle_p3$count_channel) == TRUE), 'count_channel'] = 1
test_kaggle_p3 <- test_kaggle_p3[, c('app', 'device', 'os', 'channel', 'click_id', 'count_app',
                                     'count_device', 'count_os', 'count_channel', 'count_ip', 'range_hou
# Realizando a previsão
pred_rf_v4_p3 <- predict(modelo_rf_v7, test_kaggle_p3, type="response")
# Criando um data set com o real e o previsto
prediction4_p3 <- data.frame('click_id' = test_kaggle_p3$click_id,
                             'is_attributed' = pred_rf_v4_p3)
# Visualizando os primeiros dados
head(prediction4_p3)
```

```
##   click_id is_attributed
## 1  9395233             0
## 2  9395235             0
## 3  9395236             0
## 4  9395237             0
## 5  9395238             0
## 6  9395239             0
```

```
# Excluindo o data set de teste
rm(test_kaggle_p3)


# Lendo o arquivo parte 4
test_kaggle_p4 <- fread("https://media.githubusercontent.com/media/fthara/AdTracking_Fraude_Detection_c
test_kaggle_p4 <- transform_test(test_kaggle_p4)
# Criando a coluna count_app
test_kaggle_p4$count_app <- dfapp$count[match(test_kaggle_p4$app, dfapp$app)]
# Renumerando os valores NA`s para 1
test_kaggle_p4[(is.na(test_kaggle_p4$count_app) == TRUE), 'count_app'] = 1
# Criando a coluna count_device
test_kaggle_p4$count_device <- dfdevice$count[match(test_kaggle_p4$device, dfdevice$device)]
# Renumerando os valores NA`s para 1
test_kaggle_p4[(is.na(test_kaggle_p4$count_device) == TRUE), 'count_device'] = 1
# Criando a coluna count_os
```

```
test_kaggle_p4$count_os <- dfos$count[match(test_kaggle_p4$os, dfos$os)]
# Renumerando os valores NA`s para 1
test_kaggle_p4[(is.na(test_kaggle_p4$count_os) == TRUE), 'count_os'] = 1
# Criando a coluna count_app
test_kaggle_p4$count_channel <- dfchannel$count[match(test_kaggle_p4$channel, dfchannel$channel)]
# Renumerando os valores NA`s para 1
test_kaggle_p4[(is.na(test_kaggle_p4$count_channel) == TRUE), 'count_channel'] = 1
test_kaggle_p4 <- test_kaggle_p4[, c('app', 'device', 'os', 'channel', 'click_id', 'count_app',
                                     'count_device', 'count_os', 'count_channel', 'count_ip', 'range_hou
# Realizando a previsão
pred_rf_v4_p4 <- predict(modelo_rf_v7, test_kaggle_p4, type="response")
# Criando um data set com o real e o previsto
prediction4_p4 <- data.frame('click_id' = test_kaggle_p4$click_id,
                             'is_attributed' = pred_rf_v4_p4)
# Visualizando os primeiros dados
head(prediction4_p4)
```

```
##   click_id is_attributed
## 1 14092851             0
## 2 14092852             0
## 3 14092853             0
## 4 14092854             0
## 5 14092855             0
## 6 14092856             0
```

```
# Excluindo o data set de teste
rm(test_kaggle_p4)

# Juntando as 4 predições
prediction4 <- rbind(prediction4_p1, prediction4_p2, prediction4_p3, prediction4_p4)
# Escrevendo o data set em um arquivo csv.
#fwrite(prediction4, "prediction4.csv", row.names = F, sep = ",")
```

Infelizmente esse modelo não trouxe resultados muitos relevantes para o kaggle, mas podemos ver a melhora pela métrica de AUC do modelo. Talvez treinando um pouco mais ou usando o data set original, que fica muito pesado em meu computador, o modelo fique melhor.

Conclusão

Apesar dos resultados não terem sido muito satisfatório podemos ver uma grande melhora do primeiro modelo preditivo (baseline) para os outros, isto devido ao balanceamento dos dados. A curva AUC é uma medida bem interessante para ser usada quando o assunto se refere a fraudes, pois normalmente os eventos ficam bem desbalanceados e eles precisam do balanceamento para o modelo conseguir aprender bem em relação aos dois tipos.

Fim

Fernando Tsutomu Hara