Projeto\_1\_Deteccao\_fraudes

Rafael Schena

20/11/2020

# Fromação Cientista de Dados - Projeto 1

## Big Data Analytics com R e Microsoft Azure Machine Learning

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

### Definição do problema de negócio

A TalkingData (<https://www.talkingdata.com>), a maior plataforma de Big Data independente da China, cobre mais de 70% dos dispositivos móveis ativos em todo o país. Eles lidam com 3 bilhões de cliques por dia, dos quais 90% são potencialmente fraudulentos.

O objetivo deste trabalho é construir um modelo de machine learning que possa determinar se um clique é fraudulento ou não com base em dados históricos fornecidos pela empresa e disponíveis publicamente em (<https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data>)

### Análise exploratória dos dados

Descrição dos dados fornecidos:

File descriptions train.csv - the training set train\_sample.csv - 100,000 randomly-selected rows of training data, to inspect data before downloading full set test.csv - the test set sampleSubmission.csv - a sample submission file in the correct format UPDATE: test\_supplement.csv - This is a larger test set that was unintentionally released at the start of the competition. It is not necessary to use this data, but it is permitted to do so. The official test data is a subset of this data. Data fields Each row of the training data contains a click record, with the following features.

ip: ip address of click. app: app id for marketing. device: device type id of user mobile phone (e.g., iphone 6 plus, iphone 7, huawei mate 7, etc.) os: os version id of user mobile phone channel: channel id of mobile ad publisher click\_time: timestamp of click (UTC) attributed\_time: if user download the app for after clicking an ad, this is the time of the app download is\_attributed: the target that is to be predicted, indicating the app was downloaded Note that ip, app, device, os, and channel are encoded.

The test data is similar, with the following differences:

click\_id: reference for making predictions is\_attributed: not included

##################################################################  
#### Projeto 1 - Detecção de fraudes - TalkingData AdTracking ####  
##################################################################  
  
setwd("C:/DataScience/FCD/BigDataAnalytics-R-Azure/Projeto-1/")  
getwd()

## [1] "C:/DataScience/FCD/BigDataAnalytics-R-Azure/Projeto-1"

############################################################  
#### Carregamento de dados e bibliotecas ###################  
############################################################  
  
library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

library(ggplot2)  
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.5.3

## 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

library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.5.3

library(data.table)

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

library(e1071)

## Warning: package 'e1071' was built under R version 3.5.3

# Utilizando para testes e desenvolvimento na máquina local o arquivo train\_sample.csv  
# por motivos de menor utilização de memória da máquina. Na versão para produção, será utilizado  
# o arquivo completo train.csv  
  
dados<- read.csv("train\_sample.csv")  
  
head(dados)

## 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

##############################################  
#### Análise Exploratória dos Dados ##########  
##############################################  
  
# Limpeza e manipulação  
  
# Verificando a existência de NAs no arquivo  
variaveis <- names(dados)  
  
  
for(v in variaveis){  
 print('-------------')  
 print(paste("Número de NAs na variável", v))  
 print(sum(is.na(dados[v])))  
}

## [1] "-------------"  
## [1] "Número de NAs na variável ip"  
## [1] 0  
## [1] "-------------"  
## [1] "Número de NAs na variável app"  
## [1] 0  
## [1] "-------------"  
## [1] "Número de NAs na variável device"  
## [1] 0  
## [1] "-------------"  
## [1] "Número de NAs na variável os"  
## [1] 0  
## [1] "-------------"  
## [1] "Número de NAs na variável channel"  
## [1] 0  
## [1] "-------------"  
## [1] "Número de NAs na variável click\_time"  
## [1] 0  
## [1] "-------------"  
## [1] "Número de NAs na variável attributed\_time"  
## [1] 0  
## [1] "-------------"  
## [1] "Número de NAs na variável is\_attributed"  
## [1] 0

str(dados)

## '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 : Factor w/ 80350 levels "2017-11-06 16:00:00",..: 17416 23124 27845 11509 70546 59756 59657 18083 43782 48178 ...  
## $ attributed\_time: Factor w/ 228 levels "","2017-11-06 17:19:04",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ is\_attributed : int 0 0 0 0 0 0 0 0 0 0 ...

dados\_filt <- dados %>% filter(attributed\_time != '' & is\_attributed != 1)  
head(dados\_filt)

## [1] ip app device os   
## [5] channel click\_time attributed\_time is\_attributed   
## <0 rows> (or 0-length row.names)

dados %>%  
 group\_by(is\_attributed) %>%  
 summarise(Vazio = sum(attributed\_time == ''),  
 Alguma\_coisa = sum(attributed\_time != ''))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 3  
## is\_attributed Vazio Alguma\_coisa  
## <int> <int> <int>  
## 1 0 99773 0  
## 2 1 0 227

dados$attributed\_time <- NULL  
rm(dados\_filt)  
  
# Tratando a variável click\_time: pode-se avaliar o tempo tanto como uma variável contínua,  
# no sentido de se analisar uma tendência de aumento do número de fraudes ao longo do tempo  
# como também de forma categorizada (por mês, dia do mês, dia da semana, hora do dia, etc.)  
# para analisar possíveis sazonalidades no número de fraudes ao longo do ano.  
  
# Para isto será criada uma variável click\_time\_posixct, que é a conversão da variável  
# click\_time para o formato POSIXct. Poderão ser criadas variáveis como rel\_click\_time,  
# que será o tempo relativo entre o menor tempo do dataset e cada tempo registrado,  
# e também variáveis categóricas como year, month, week, weekday, hour, minute, second.   
# A variável original click\_time será eliminada do dataset.  
  
dados$click\_time\_posixct <- as.POSIXct(dados$click\_time)  
dados$click\_time <- NULL  
  
t\_1st <- min(dados$click\_time\_posixct)  
t\_last <- max(dados$click\_time\_posixct)  
  
t\_last - t\_1st

## Time difference of 2.999896 days

# Como o tempo decorrido entre o primeiro e o último clique é de apenas 3 dias, não parece  
# fazer sentido um aumento do número de fraudes significativo ao longo desse curto intervalo de  
# tempo. De modo que as variáveis year, month, week, weekday não serão criadas para poupar  
# esforço computacional. Serão criadas, então, apenas as variáveis hour, minute, second.  
# Também para reduzir esforço computacional, será eliminada do dataset de treino a variável  
# click\_time\_posixct.  
  
dados$hour <- hour(dados$click\_time\_posixct)  
dados$minute <- minute(dados$click\_time\_posixct)  
dados$second <- second(dados$click\_time\_posixct)  
dados$click\_time\_posixct <- NULL  
str(dados)

## 'data.frame': 100000 obs. of 9 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 ...  
## $ is\_attributed: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ hour : int 9 13 18 4 9 1 1 10 9 12 ...  
## $ minute : int 30 40 5 58 0 22 17 1 35 35 ...  
## $ second : int 38 27 24 8 9 13 58 53 17 26 ...

summary(dados)

## ip app device os   
## Min. : 9 Min. : 1.00 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 40552 1st Qu.: 3.00 1st Qu.: 1.00 1st Qu.: 13.00   
## Median : 79827 Median : 12.00 Median : 1.00 Median : 18.00   
## Mean : 91256 Mean : 12.05 Mean : 21.77 Mean : 22.82   
## 3rd Qu.:118252 3rd Qu.: 15.00 3rd Qu.: 1.00 3rd Qu.: 19.00   
## Max. :364757 Max. :551.00 Max. :3867.00 Max. :866.00   
## channel is\_attributed hour minute   
## Min. : 3.0 Min. :0.00000 Min. : 0.000 Min. : 0.00   
## 1st Qu.:145.0 1st Qu.:0.00000 1st Qu.: 4.000 1st Qu.:14.00   
## Median :258.0 Median :0.00000 Median : 9.000 Median :29.00   
## Mean :268.8 Mean :0.00227 Mean : 9.329 Mean :29.16   
## 3rd Qu.:379.0 3rd Qu.:0.00000 3rd Qu.:14.000 3rd Qu.:44.00   
## Max. :498.0 Max. :1.00000 Max. :23.000 Max. :59.00   
## second   
## Min. : 0.00   
## 1st Qu.:15.00   
## Median :30.00   
## Mean :29.78   
## 3rd Qu.:45.00   
## Max. :59.00

Não existem dados missing no conjunto de dados.

O dataset apresenta 7 atributos, sendo 2 do tipo fator e 5 do tipo inteiro.

A princípio, não faz sentido haver atributos do tipo inteiro, uma vez que eles não descrevem quantidades, mas sim categorias. Em testes preliminares, o algoritmo randomForest apresentou uma limitação em processar atributos com mais de 53 categorias. Serão construídos modelos de machine learning tanto com os dados fatorizados quanto com os dados nos formatos originais e serão avaliados os desempenhos.

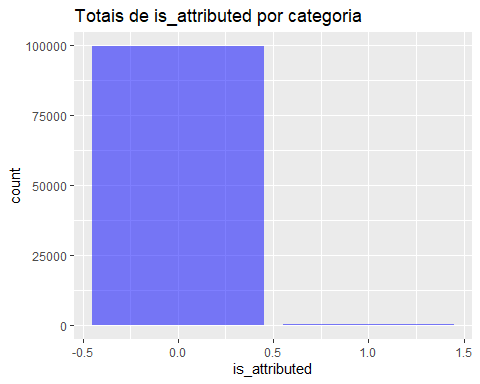
A variável attributed\_time, uma das variáveis tipo fator, não traz informação alguma, uma vez que ela só é preenchida quando o target é igual a 1, e nem está presente nos dados de teste. Assim sendo, foi excluída do dataset.

O outro atributo tipo fator (click\_time) pode ser avaliada tanto como uma variável contínua no sentido de se analisar uma tendência de aumento do número de fraudes ao longo do tempo como também de forma categorizada (por mês, dia do mês, dia da semana, hora do dia, etc.) para analisar possíveis sazonalidades no número de fraudes ao longo do ano.

Para isto foi criada uma variável click\_time\_posixct, que é a conversão da variável click\_time para o formato POSIXct. O intuito foi criar variáveis categóricas como year, month, week, weekday, hour, minute, second. A variável original click\_time foi também eliminada do dataset.

Como o tempo decorrido entre o primeiro e o último clique é de apenas 3 dias, não parece fazer sentido um aumento do número de fraudes significativo ao longo desse curto intervalo de tempo. De modo que as variáveis year, month, week, weekday não foram criadas. Foram criadas, então, apenas as variáveis hour, minute, second. Após feitas estas manipulações, a variável click\_time\_posixct foi excluída para eliminar informações redundantes.

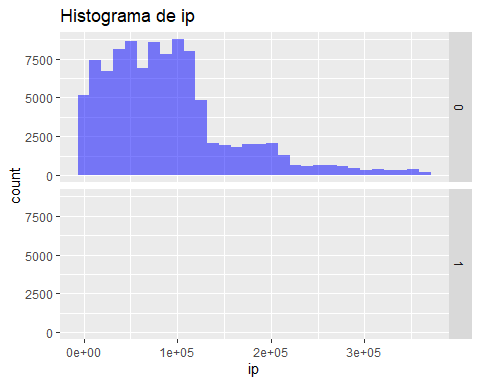
Próximo passo a ser realizado é uma análise exploratória dos dados.



##   
## 0 1   
## 99773 227

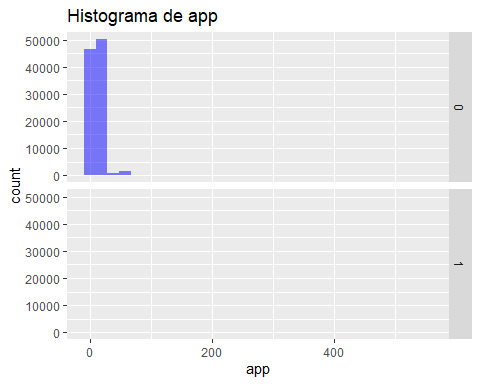
## [[1]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



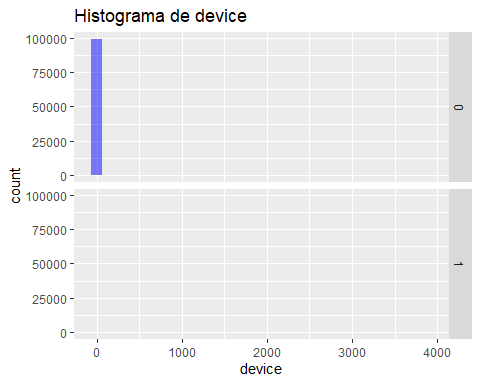
##   
## [[2]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



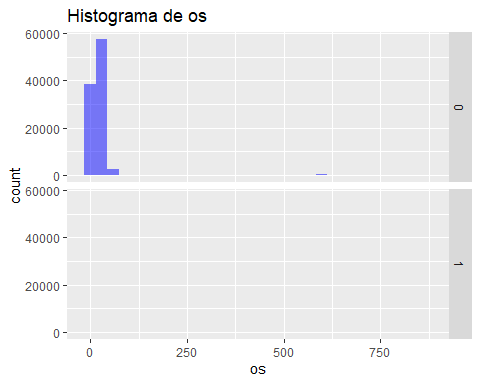
##   
## [[3]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



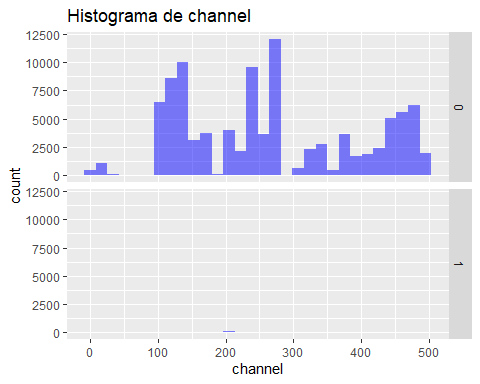
##   
## [[4]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



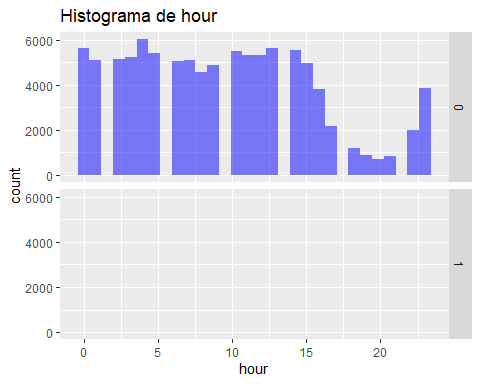
##   
## [[5]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



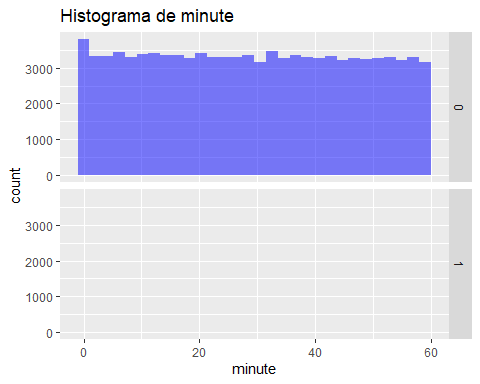
##   
## [[6]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



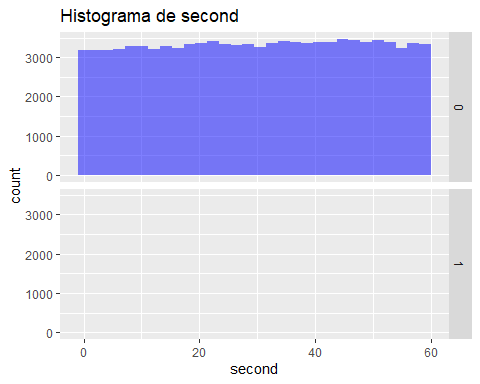
##   
## [[7]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



##   
## [[8]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

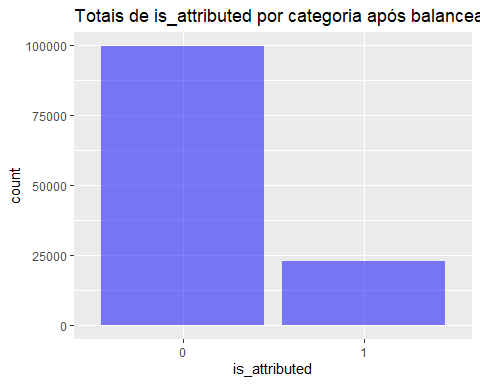
 Constata-se que os dados desbalanceados de tal modo que torna impossível a avaliação de quaisquer relações entre as variáveis de entrada e o rótulo atribuído a cada observação.

Tal desbalanceamento também é prejudicial à construção de modelos de machine learning, de forma que deve ser adotada alguma técnica de balanceamento.

Para contornar este problema foi utilizada a técnica SMOTE (Synthetic Minority Oversampling Technique), que vai sintetizar dados da menor categoria. Estabeleceu-se como parâmetro que a menor categoria seja aumentada para um total de aproximadamente 20% do dataset. Como originalmente o dataset contém 227 observações positivas em um total de 100.000 observações (ou seja, 0,227%), é necessário aumentar o número de observações positivas em uma razão de 100, ou 10.000%.

Foi utilizado o Azure Machine Learning para balanceamento do dataset.

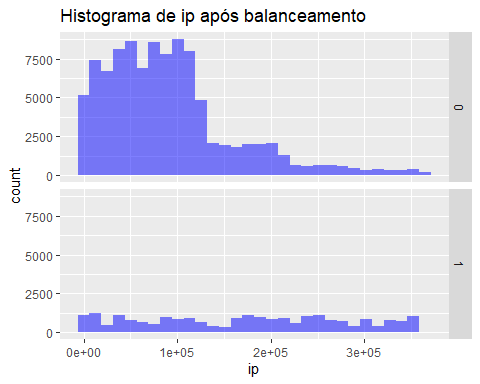
## 'data.frame': 122700 obs. of 9 variables:  
## $ ip : int 87540 105560 101424 94584 68413 93663 17059 121505 192967 143636 ...  
## $ app : int 12 25 12 13 12 3 1 9 2 3 ...  
## $ device : int 1 1 1 1 1 1 1 1 2 1 ...  
## $ os : int 13 17 19 13 1 17 17 25 22 19 ...  
## $ channel : int 497 259 212 477 178 115 135 442 364 135 ...  
## $ is\_attributed: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ hour : int 9 13 18 4 9 1 1 10 9 12 ...  
## $ minute : int 30 40 5 58 0 22 17 1 35 35 ...  
## $ second : int 38 27 24 8 9 13 58 53 17 26 ...



##   
## 0 1   
## 99773 22927

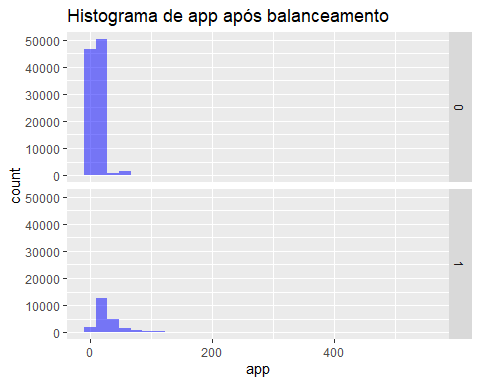
## [[1]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



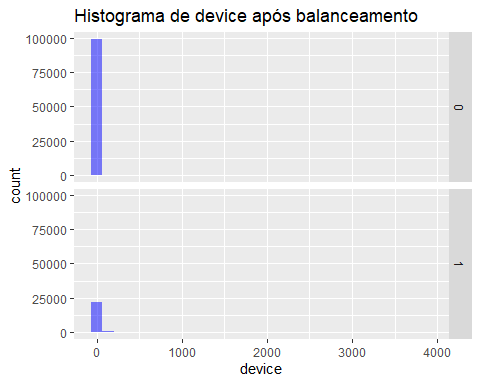
##   
## [[2]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



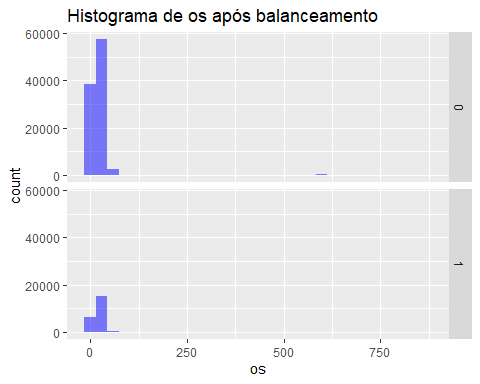
##   
## [[3]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



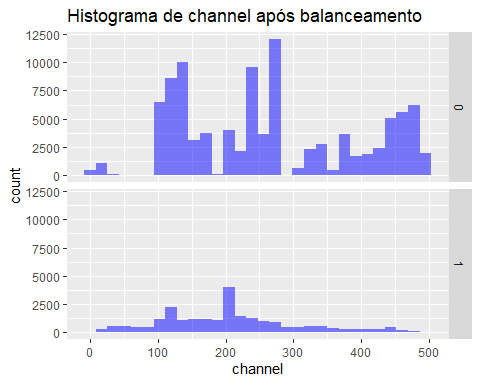
##   
## [[4]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



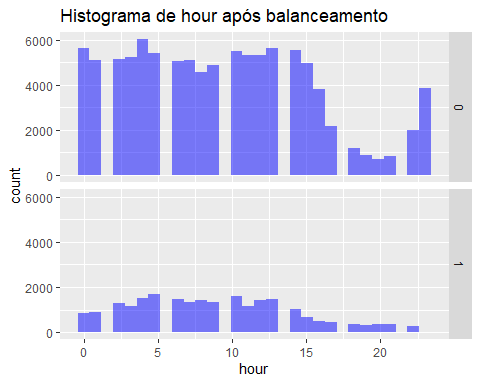
##   
## [[5]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



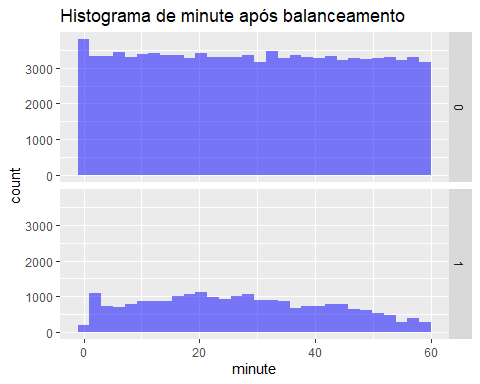
##   
## [[6]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



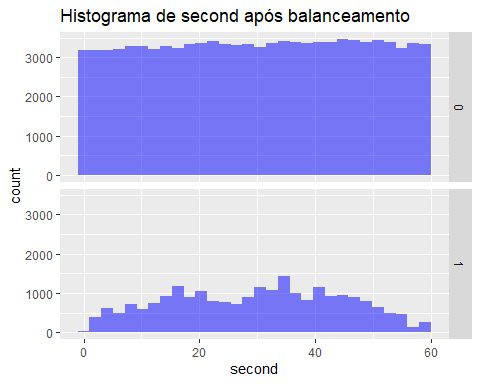
##   
## [[7]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



##   
## [[8]]

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Visualmente nota-se padrões diferentes de distribuição nos histogramas para os atributos ip, app, channel, minute e second.

Parte-se agora para a construção e testes dos modelos de machine learning, onde serão utilizados 2 tipos de algoritmos: randomForest e SVM. Ambos os algoritmos serão testados com os atributos do tipo inteiro, como estavam originalmente, e também categorizados.

## Criando um dataset categorizado para posterior teste dos modelos  
variaveis <- names(dados)  
dados\_cat <- dados  
  
for(v in variaveis){  
 dados\_cat[[v]] <- factor(dados\_cat[[v]])  
}  
str(dados\_cat)

## 'data.frame': 122700 obs. of 9 variables:  
## $ ip : Factor w/ 50302 levels "9","10","19",..: 19156 23222 22177 20762 15079 20405 3921 26705 36314 29659 ...  
## $ app : Factor w/ 248 levels "1","2","3","4",..: 12 25 12 13 12 3 1 9 2 3 ...  
## $ device : Factor w/ 473 levels "0","1","2","3",..: 2 2 2 2 2 2 2 2 3 2 ...  
## $ os : Factor w/ 455 levels "0","1","2","3",..: 14 18 20 14 2 18 18 26 23 20 ...  
## $ channel : Factor w/ 491 levels "3","4","5","6",..: 490 257 210 475 176 113 133 440 362 133 ...  
## $ is\_attributed: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ hour : Factor w/ 24 levels "0","1","2","3",..: 10 14 19 5 10 2 2 11 10 13 ...  
## $ minute : Factor w/ 60 levels "0","1","2","3",..: 31 41 6 59 1 23 18 2 36 36 ...  
## $ second : Factor w/ 60 levels "0","1","2","3",..: 39 28 25 9 10 14 59 54 18 27 ...

Percebe-se que numa primeira tentativa, os atributos ip, app, device, os, channel, minute e second extrapolam o número máximo de 53 níveis do randomForest. Serão utilizados 53 níveis para estas variáveis.

dados\_cat <- dados  
variaveis <- names(dados)  
variaveis <- variaveis[!(variaveis %in% c("is\_attributed", "hour"))]  
  
for(v in variaveis){  
 dados\_cat[[v]] <- cut(dados[[v]], 53)  
}  
  
# Categorizando também o atributo hour, que tem menos que 53 níveis  
dados\_cat$hour <- factor(dados\_cat$hour)  
  
str(dados\_cat)

## 'data.frame': 122700 obs. of 9 variables:  
## $ ip : Factor w/ 53 levels "(-356,6.89e+03]",..: 13 16 15 14 10 14 3 18 29 21 ...  
## $ app : Factor w/ 53 levels "(0.45,11.4]",..: 2 3 2 2 2 1 1 1 1 1 ...  
## $ device : Factor w/ 53 levels "(-3.87,73]","(73,146]",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ os : Factor w/ 53 levels "(-0.866,16.3]",..: 1 2 2 1 1 2 2 2 2 2 ...  
## $ channel : Factor w/ 53 levels "(2.5,12.3]","(12.3,21.7]",..: 53 28 23 51 19 12 15 48 39 15 ...  
## $ is\_attributed: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ hour : Factor w/ 24 levels "0","1","2","3",..: 10 14 19 5 10 2 2 11 10 13 ...  
## $ minute : Factor w/ 53 levels "(-0.059,1.11]",..: 27 36 5 53 1 20 16 1 32 32 ...  
## $ second : Factor w/ 53 levels "(-0.059,1.11]",..: 35 25 22 8 9 12 53 48 16 24 ...

str(dados)

## 'data.frame': 122700 obs. of 9 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 ...  
## $ is\_attributed: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ hour : int 9 13 18 4 9 1 1 10 9 12 ...  
## $ minute : int 30 40 5 58 0 22 17 1 35 35 ...  
## $ second : int 38 27 24 8 9 13 58 53 17 26 ...

### Construção dos modelos de machine learning

Antes do treinamento e avaliação dos modelos é necessário dividir os dados em dados de treino e validação, uma vez que os dados de teste fornecidos não têm label.

# Fazendo o split data entre dados de treino e dados de validação, uma vez que os dados  
# de teste não tem label disponível.  
  
split <- createDataPartition(y = dados$is\_attributed, p = 0.7, list = FALSE)  
dados\_treino <- dados[split, ]  
dados\_valid <- dados[-split, ]  
  
dados\_cat\_treino <- dados\_cat[split, ]  
dados\_cat\_valid <- dados\_cat[-split, ]

#### Primeiro modelo de machine learning

Algoritmo: randomForest Atributos: não-categorizados

###################################################  
#### Primeiro modelo de Machine Learning ##########  
###################################################  
# Algoritmo: randomForest  
# Atributos: não-categorizados  
  
modelo\_rf1 <- randomForest(is\_attributed ~ .,  
 data = dados\_treino,  
 ntree = 100, nodesize = 10, importance = T)  
  
previsao\_rf1 <- predict(modelo\_rf1, dados\_valid, type = 'class')  
  
  
# Matriz de confusão  
confusionMatrix(previsao\_rf1, dados\_valid$is\_attributed)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 29868 214  
## 1 63 6664  
##   
## Accuracy : 0.9925   
## 95% CI : (0.9915, 0.9933)  
## No Information Rate : 0.8131   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.975   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9979   
## Specificity : 0.9689   
## Pos Pred Value : 0.9929   
## Neg Pred Value : 0.9906   
## Prevalence : 0.8131   
## Detection Rate : 0.8114   
## Detection Prevalence : 0.8172   
## Balanced Accuracy : 0.9834   
##   
## 'Positive' Class : 0   
##

#### Segundo modelo de machine learning

Algoritmo: randomForest Atributos: categorizados

###################################################  
#### Segundo modelo de Machine Learning ###########  
###################################################  
# Algoritmo: randomForest  
# Atributos: categorizados  
  
modelo\_rf2 <- randomForest(is\_attributed ~ .,  
 data = dados\_cat\_treino,  
 ntree = 100, nodesize = 10, importance = T)  
  
previsao\_rf2 <- predict(modelo\_rf2, dados\_cat\_valid, type = 'class')  
  
  
  
# Matriz de confusão  
confusionMatrix(previsao\_rf2, dados\_cat\_valid$is\_attributed)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 29463 753  
## 1 468 6125  
##   
## Accuracy : 0.9668   
## 95% CI : (0.9649, 0.9686)  
## No Information Rate : 0.8131   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8891   
##   
## Mcnemar's Test P-Value : 4.38e-16   
##   
## Sensitivity : 0.9844   
## Specificity : 0.8905   
## Pos Pred Value : 0.9751   
## Neg Pred Value : 0.9290   
## Prevalence : 0.8131   
## Detection Rate : 0.8004   
## Detection Prevalence : 0.8209   
## Balanced Accuracy : 0.9374   
##   
## 'Positive' Class : 0   
##

#### Terceiro modelo de machine learning

Algoritmo: SVM Atributos: não-categorizados

###################################################  
#### Terceiro modelo de Machine Learning ##########  
###################################################  
# Algoritmo: SVM  
# Atributos: não-categorizados  
  
modelo\_svm1 <- svm(is\_attributed ~ .,  
 data = dados\_treino,  
 type = 'C-classification',  
 kernel = 'radial')  
  
previsao\_svm1 <- predict(modelo\_svm1, dados\_valid, type = 'class')  
  
  
# Matriz de confusão  
confusionMatrix(previsao\_svm1, dados\_valid$is\_attributed)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 29194 2168  
## 1 737 4710  
##   
## Accuracy : 0.9211   
## 95% CI : (0.9183, 0.9238)  
## No Information Rate : 0.8131   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7177   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9754   
## Specificity : 0.6848   
## Pos Pred Value : 0.9309   
## Neg Pred Value : 0.8647   
## Prevalence : 0.8131   
## Detection Rate : 0.7931   
## Detection Prevalence : 0.8520   
## Balanced Accuracy : 0.8301   
##   
## 'Positive' Class : 0   
##

#### Quarto modelo de machine learning

Algoritmo: SVM Atributos: categorizados

###################################################  
#### Quarto modelo de Machine Learning ############  
###################################################  
# Algoritmo: SVM  
# Atributos: categorizados  
  
modelo\_svm2 <- svm(is\_attributed ~ .,  
 data = dados\_cat\_treino,  
 type = 'C-classification',  
 kernel = 'radial')  
  
previsao\_svm2 <- predict(modelo\_svm2, dados\_cat\_valid, type = 'class')  
  
  
# Matriz de confusão  
confusionMatrix(previsao\_svm2, dados\_cat\_valid$is\_attributed)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 29668 3196  
## 1 263 3682  
##   
## Accuracy : 0.906   
## 95% CI : (0.903, 0.909)  
## No Information Rate : 0.8131   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.63   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9912   
## Specificity : 0.5353   
## Pos Pred Value : 0.9028   
## Neg Pred Value : 0.9333   
## Prevalence : 0.8131   
## Detection Rate : 0.8060   
## Detection Prevalence : 0.8928   
## Balanced Accuracy : 0.7633   
##   
## 'Positive' Class : 0   
##

#### Quinto modelo de machine learning

Algoritmo: Naive-Bayes Atributos: não-categorizados

###################################################  
#### Quinto modelo de Machine Learning ############  
###################################################  
# Algoritmo: Naive-Bayes  
# Atributos: não-categorizados  
  
modelo\_nb1 <- naiveBayes(dados[, -6], dados[, 6])  
  
  
previsao\_nb1 <- predict(modelo\_nb1, dados\_valid, type = 'class')  
  
  
# Matriz de confusão  
confusionMatrix(previsao\_svm1, dados\_valid$is\_attributed)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 29194 2168  
## 1 737 4710  
##   
## Accuracy : 0.9211   
## 95% CI : (0.9183, 0.9238)  
## No Information Rate : 0.8131   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7177   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9754   
## Specificity : 0.6848   
## Pos Pred Value : 0.9309   
## Neg Pred Value : 0.8647   
## Prevalence : 0.8131   
## Detection Rate : 0.7931   
## Detection Prevalence : 0.8520   
## Balanced Accuracy : 0.8301   
##   
## 'Positive' Class : 0   
##

#### Sexto modelo de machine learning

Algoritmo: Naive-Bayes Atributos: categorizados

###################################################  
#### Sexto modelo de Machine Learning #############  
###################################################  
# Algoritmo: Naive-Bayes  
# Atributos: categorizados  
  
modelo\_nb2 <- naiveBayes(dados\_cat[, -6], dados[, 6])  
  
  
previsao\_nb2 <- predict(modelo\_nb2, dados\_cat\_valid, type = 'class')  
  
  
# Matriz de confusão  
confusionMatrix(previsao\_svm1, dados\_valid$is\_attributed)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 29194 2168  
## 1 737 4710  
##   
## Accuracy : 0.9211   
## 95% CI : (0.9183, 0.9238)  
## No Information Rate : 0.8131   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7177   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9754   
## Specificity : 0.6848   
## Pos Pred Value : 0.9309   
## Neg Pred Value : 0.8647   
## Prevalence : 0.8131   
## Detection Rate : 0.7931   
## Detection Prevalence : 0.8520   
## Balanced Accuracy : 0.8301   
##   
## 'Positive' Class : 0   
##

Melhor modelo em performance: primeiro modelo, com algoritmo randomForest, treinado com os dados não-categorizados.

Fazendo as previsões no arquivo de submissão ao Kaggle.

###################################################  
#### Predição do arquivo test.csv #################  
###################################################  
  
# Melhor modelo: modelo\_rf1  
  
test <- as.data.frame(fread("test.csv"))  
submission <- as.data.frame(fread("sample\_submission.csv"))  
head(test)

## click\_id ip app device os channel click\_time  
## 1 0 5744 9 1 3 107 2017-11-10 04:00:00  
## 2 1 119901 9 1 3 466 2017-11-10 04:00:00  
## 3 2 72287 21 1 19 128 2017-11-10 04:00:00  
## 4 3 78477 15 1 13 111 2017-11-10 04:00:00  
## 5 4 123080 12 1 13 328 2017-11-10 04:00:00  
## 6 5 110769 18 1 13 107 2017-11-10 04:00:00

head(submission)

## click\_id is\_attributed  
## 1 0 0  
## 2 1 0  
## 3 2 0  
## 4 3 0  
## 5 4 0  
## 6 5 0

str(test)

## 'data.frame': 18790469 obs. of 7 variables:  
## $ click\_id : int 0 1 2 3 4 5 6 7 9 8 ...  
## $ ip : int 5744 119901 72287 78477 123080 110769 12540 88637 14932 123701 ...  
## $ app : int 9 9 21 15 12 18 3 27 18 12 ...  
## $ device : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ os : int 3 3 19 13 13 13 1 19 10 53 ...  
## $ channel : int 107 466 128 111 328 107 137 153 107 424 ...  
## $ click\_time: chr "2017-11-10 04:00:00" "2017-11-10 04:00:00" "2017-11-10 04:00:00" "2017-11-10 04:00:00" ...

str(submission)

## 'data.frame': 18790469 obs. of 2 variables:  
## $ click\_id : int 0 1 2 3 4 5 6 7 9 8 ...  
## $ is\_attributed: int 0 0 0 0 0 0 0 0 0 0 ...

# Para adequar o dataset de teste ao dataset de treinamento do modelo 1  
# é necessário excluir a variável click\_id e formatar a variável click\_time  
test$click\_id <- NULL  
  
test$click\_time\_posixct <- as.POSIXct(test$click\_time)  
test$click\_time <- NULL  
  
  
test$hour <- hour(test$click\_time\_posixct)  
test$minute <- minute(test$click\_time\_posixct)  
test$second <- second(test$click\_time\_posixct)  
test$click\_time\_posixct <- NULL  
str(test)

## 'data.frame': 18790469 obs. of 8 variables:  
## $ ip : int 5744 119901 72287 78477 123080 110769 12540 88637 14932 123701 ...  
## $ app : int 9 9 21 15 12 18 3 27 18 12 ...  
## $ device : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ os : int 3 3 19 13 13 13 1 19 10 53 ...  
## $ channel: int 107 466 128 111 328 107 137 153 107 424 ...  
## $ hour : int 4 4 4 4 4 4 4 4 4 4 ...  
## $ minute : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ second : int 0 0 0 0 0 0 0 0 0 0 ...

previsao\_final <- predict(modelo\_rf1, test)  
table(previsao\_final)

## previsao\_final  
## 0 1   
## 18742975 47494

# Gravando o arquivo submission para envio para o Kaggle  
submission$is\_attributed <- previsao\_final  
fwrite(submission, "submission.csv")

### Conclusão

Devido a restrições de memória da máquina local, não foi possível a utilização do arquivo completo para treinamento dos modelos de machine learning. Dentre os modelos construídos, o modelo de melhor desempenho foi o primeiro modelo, com acurácia de 99,27%, sensitividade de 99,81% (percentual de acerto para a classe 0) e especificidade de 96,93% (percentual de acerto para a classe 1). Para todos os algoritmos testados, a fatorização dos dados de entrada piorou os desempenhos dos modelos. Também em todos os outros modelos foi observada uma queda significativa na especificidade em relação ao modelo 1, fato este que não foi observado na sensitividade, provavelmente devido ao número reduzido de dados da classe 1 disponíveis. Possíveis pontos de melhoria a serem testados poderiam ser um feature selection para otimização dos dados de entrada, bem como utilizar a penalização para erros no algoritmo random forest através do modelo C5.0.