Projeto Data Science

Aplicação de Machine Learning para Detecção de Fraudes

Anderlane Oliveira

25/08/2021

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

Pesquisadora: Anderlane Oliveira

Objetivo

Criar um modelo de aprendizado de máquina, onde seja possível prever se o usuário fará o download de um aplicativo, após o mesmo clicar em um anúncio referente a esse aplicativo para dispositivos móveis.

Etapas do Projeto

1.Carregamento e vizualização dos dados obtidos

2. Informações preliminares sobre os dados

3. Transformação e Manipulação dos Dados

4. Análise Exploratória de Dados

5. Modelagem Preditiva Sem Balanceamento do Dataset

6. Modelagem Preditiva após Balanceamento do Dataset

7. Avaliação do Modelo Preditivo

Desenvolvimento

1. Carregamento e vizualização dos dados obtidos

Dataset disponível no site Kaggle:

https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data

Descrição das variáveis:

ip: endereço ip do clique.

app: id do app para marketing.

device: ID do tipo de dispositivo do telefone celular do usuário (por exemplo, iphone 6 plus, iphone 7, huawei mate 7 etc.)

os: id da versão do sistema operacional do telefone celular do usuário

channel: id do canal do editor de anúncios para celular

click_time: carimbo de data / hora do clique (UTC)

attributed_time: se o usuário baixar o aplicativo depois de clicar em um anúncio, é o momento do download do aplicativo

is_attributed: o destino que deve ser previsto, indicando que o aplicativo foi baixado

Observação: As variáveis ip, app, device, os e channel são codificados.

```
# Definindo o diretório de trabalho
setwd("C:/FCD/BigDataRAzure/ProjetoFinal01")


# Carregando os pacotes
library(dplyr) # Manipular os dados
```

```
##
## 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(corrplot) # Criar matrix de correlação
```

```
## corrplot 0.89 loaded

library(ggplot2) # Criar gráficos
library(class) # Criar modelo preditivo
```

Carregando o dataset

```
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
```

2. Informações preliminares sobre os dados

```
# Informações sobre os dados

str(dados)

## 'data.frame':    100000 obs. of  8 variables:
##  $ ip             : int  87540 105560 101424 94584 68413 93663 17059 121505 1929
67 143636 ...
##  $ app            : int  12 25 12 13 12 3 1 9 2 3 ...
##  $ device         : int  1 1 1 1 1 1 1 1 2 1 ...
##  $ os             : int  13 17 19 13 1 17 17 25 22 19 ...
##  $ channel        : int  497 259 212 477 178 115 135 442 364 135 ...
##  $ click_time     : chr  "2017-11-07 09:30:38" "2017-11-07 13:40:27" "2017-11-07
18:05:24" "2017-11-07 04:58:08" ...
##  $ attributed_time: chr  "" "" "" "" ...
##  $ is_attributed  : int  0 0 0 0 0 0 0 0 0 0 ...

dim(dados)

## [1] 100000      8

class(dados)

## [1] "data.frame"

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       click_time        attributed_time    is_attributed     
##  Min.   :  3.0   Length:100000      Length:100000      Min.   :0.00000  
##  1st Qu.:145.0   Class :character   Class :character   1st Qu.:0.00000  
##  Median :258.0   Mode  :character   Mode  :character   Median :0.00000  
##  Mean   :268.8                                         Mean   :0.00227  
##  3rd Qu.:379.0                                         3rd Qu.:0.00000  
##  Max.   :498.0                                         Max.   :1.00000
```

Verificando dados missing no dataset

```
sum(is.na(dados))
```

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

Utilizando Tabelas de Contigência para verificação preliminar da variável alvo

```
# Quantitativo da variável alvo
table(dados$is_attributed)
```

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

3. Transformação e Manipulação dos Dados

Formatando colunas codificadas como variáveis categóricas

```
# Função para transformar variáveis para o tipo fator
transf_factor = function(df, variavel){
  for (variavel in variavel){
    df[[variavel]] <- as.factor(df[[variavel]])
  }
  return(df)
}

# Lista de variáveis
cat.var <- c('ip', 'app', 'device', 'os', 'channel', 'is_attributed')

# Transformando os dados
dados <- transf_factor(dados, cat.var)
```

Formatando as colunas de data

```
# Formatando variáveis data com POSIXct
dados$click_time <- as.POSIXct(dados$click_time, format = "%Y-%m-%d %H:%M:%S")
```

```
dados$attributed_time <- as.POSIXct(dados$click_time, format = "%Y-%m-%d %H:%M:%S")

# Vizualização do resultado após transformações realizadas
str(dados)
```

```
## 'data.frame':    100000 obs. of  8 variables:
##  $ ip             : Factor w/ 34857 levels "9","10","19",..: 15221 18449 17664 1
6497 11853 16301 2974 21304 28060 23645 ...
##  $ app            : Factor w/ 161 levels "1","2","3","4",..: 12 25 12 13 12 3 1
9 2 3 ...
##  $ device         : Factor w/ 100 levels "0","1","2","4",..: 2 2 2 2 2 2 2 3 2
...
##  $ os             : Factor w/ 130 levels "0","1","2","3",..: 14 18 20 14 2 18 18
26 23 20 ...
##  $ channel        : Factor w/ 161 levels "3","4","5","13",..: 160 68 53 147 46 2
1 35 127 101 35 ...
##  $ click_time     : POSIXct, format: "2017-11-07 09:30:38" "2017-11-07 13:40:27"
...
##  $ attributed_time: POSIXct, format: "2017-11-07 09:30:38" "2017-11-07 13:40:27"
...
##  $ is_attributed  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 ...
```

```
summary(dados)
```

```
##        ip             app            device            os
##  5348   :  669   3      :18279   1      :94338   19     :23870
##  5314   :  616   12     :13198   2      : 4345   13     :21223
##  73487  :  439   2      :11737   0      :  541   17     : 5232
##  73516  :  399   9      : 8992   3032   :  371   18     : 4830
##  53454  :  280   15     : 8595   3543   :  151   22     : 4039
##  114276 :  219   18     : 8315   3866   :   93   10     : 2816
##  (Other):97378   (Other):30884   (Other):  161   (Other):37990
##     channel        click_time                  attributed_time
##  280    : 8114   Min.   :2017-11-06 16:00:00   Min.   :2017-11-06 16:00:00
##  245    : 4802   1st Qu.:2017-11-07 11:34:09   1st Qu.:2017-11-07 11:34:09
##  107    : 4543   Median :2017-11-08 07:07:50   Median :2017-11-08 07:07:50
##  477    : 3960   Mean   :2017-11-08 06:29:52   Mean   :2017-11-08 06:29:52
##  134    : 3224   3rd Qu.:2017-11-09 02:06:01   3rd Qu.:2017-11-09 02:06:01
##  259    : 3130   Max.   :2017-11-09 15:59:51   Max.   :2017-11-09 15:59:51
##  (Other):72227
##  is_attributed
##  0:99773
##  1:  227
##
##
##
##
##
```

4. Análise Exploratória dos Dados

Vizualização dos dados em diferentes perspectivas

```
# Verificando o quantitativo por IP
dados %>%
  filter(is_attributed == 0) %>%
  group_by(ip) %>%
  summarise(total = n(), perc_total = n() / 99773 * 100) %>%
  arrange(desc(total))
```

```
## # A tibble: 34,707 x 3
##    ip     total perc_total
##    <fct>  <int>      <dbl>
##  1 5348     666      0.668
##  2 5314     613      0.614
##  3 73487    439      0.440
##  4 73516    399      0.400
##  5 53454    280      0.281
##  6 114276   219      0.219
##  7 26995    218      0.218
##  8 95766    205      0.205
##  9 17149    186      0.186
## 10 100275   173      0.173
## # ... with 34,697 more rows
```

```
# Verificando o quantitativo por APP
dados %>%
  filter(is_attributed == 0) %>%
  group_by(app) %>%
  summarise(total = n(), perc_total = n() / 99773 * 100) %>%
  arrange(desc(total))
```

```
## # A tibble: 155 x 3
##    app   total perc_total
##    <fct> <int>      <dbl>
##  1 3     18275      18.3
##  2 12    13197      13.2
##  3 2     11737      11.8
##  4 9      8984       9.00
##  5 15     8593       8.61
##  6 18     8310       8.33
##  7 14     5359       5.37
##  8 1      3135       3.14
##  9 13     2422       2.43
## 10 8      2000       2.00
## # ... with 145 more rows
```

```
# Verificando o quantitativo pela DATA
dados %>%
  filter(is_attributed == 0) %>%
```

```
  group_by(click_time) %>%
  summarise(total = n(), perc_total = n() / 99773 * 100) %>%
  arrange(desc(total))
```

```
## # A tibble: 80,218 x 3
##    click_time          total perc_total
##    <dttm>              <int>      <dbl>
##  1 2017-11-08 12:01:02     7    0.00702
##  2 2017-11-07 04:36:16     6    0.00601
##  3 2017-11-07 05:00:11     6    0.00601
##  4 2017-11-08 13:32:05     6    0.00601
##  5 2017-11-09 14:46:23     6    0.00601
##  6 2017-11-06 23:27:07     5    0.00501
##  7 2017-11-07 00:14:03     5    0.00501
##  8 2017-11-07 00:43:13     5    0.00501
##  9 2017-11-07 06:28:49     5    0.00501
## 10 2017-11-07 08:32:43     5    0.00501
## # ... with 80,208 more rows
```

```
# Agrupando o quantitativo por APP e IP
dados %>%
  filter(is_attributed == 0) %>%
  group_by(app, ip) %>%
  summarise(total = n(), perc_total = n() / 99773 * 100) %>%
  arrange(desc(total))
```

```
## `summarise()` has grouped output by 'app'. You can override using the `.groups`
argument.
```

```
## # A tibble: 76,069 x 4
## # Groups:   app [155]
##    app   ip    total perc_total
##    <fct> <fct> <int>      <dbl>
##  1 12    73487   132     0.132
##  2 3     5348    117     0.117
##  3 3     73487   104     0.104
##  4 12    73516    96     0.0962
##  5 3     73516    90     0.0902
##  6 3     5314     88     0.0882
##  7 12    5314     70     0.0702
##  8 12    5348     68     0.0682
##  9 9     5348     67     0.0672
## 10 15    5348     64     0.0641
## # ... with 76,059 more rows
```

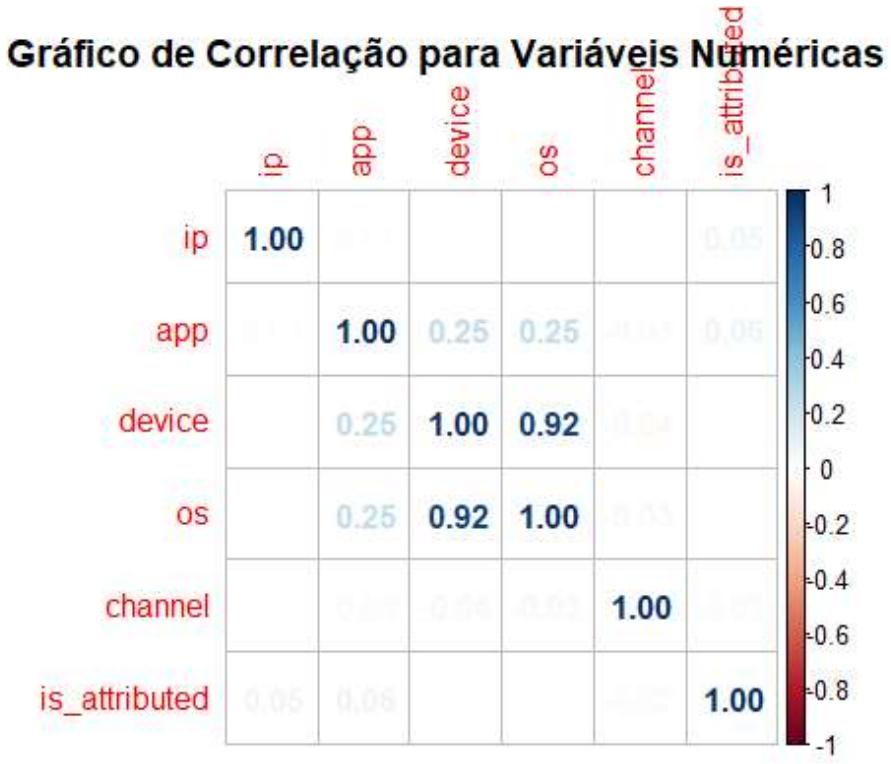
Observa-se que alguns IP's são responsáveis por grandes quantidades de cliques nos anúncios dentro do período pesquisado.

Os APP's do tipo 3, 12 e 2 correspondem a 43,3% dos acessos que não realizam o dowload indicado nos anúncios.

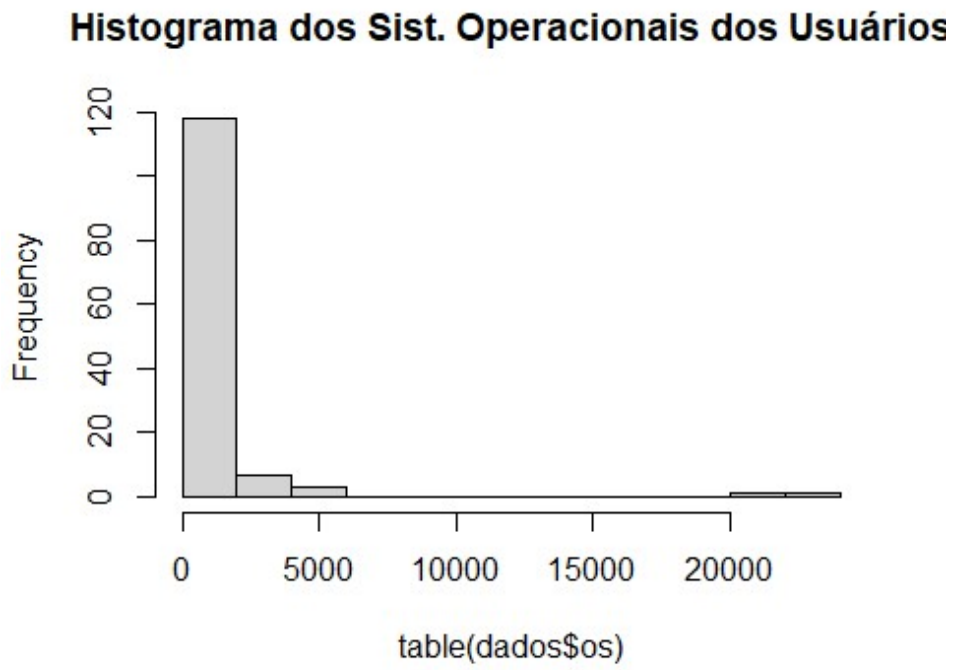
Os horários de acesso estão distribuidos ao longo do período observado.

Gráficos

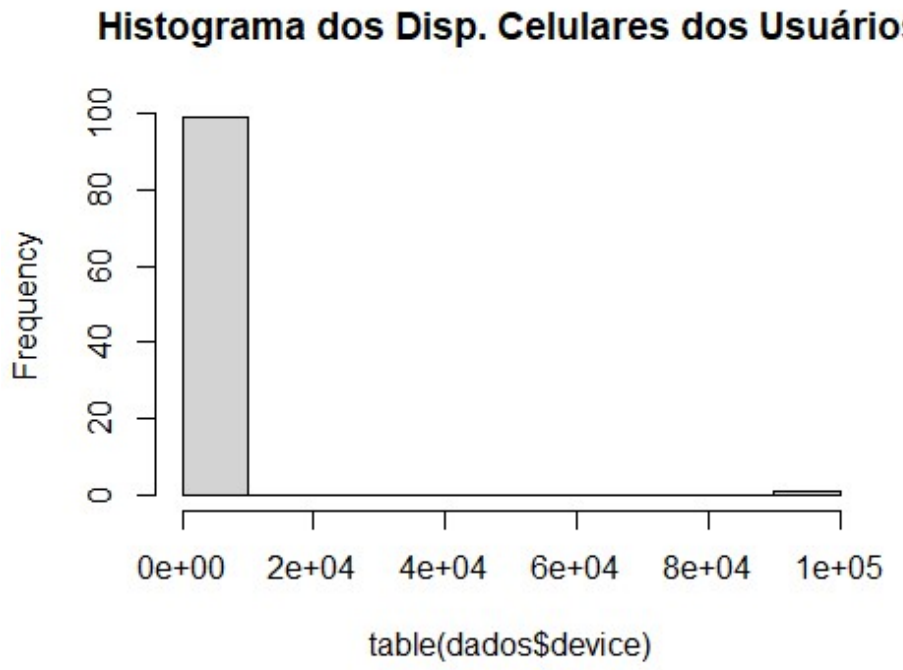
```
# Correlação entre variáveis
dados1 <- read.csv("train_sample.csv")
var_num <- sapply(dados1, is.numeric)
corr.matrix <- cor(dados1[,var_num])
corrplot(corr.matrix, main="\n\nGráfico de Correlação para Variáveis Numéricas", method="number")
```



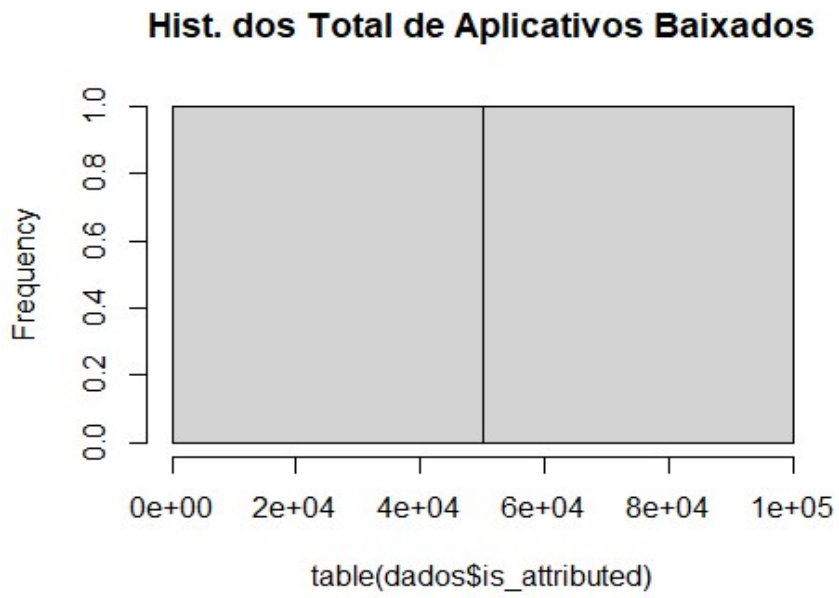

```
# Histograma
hist(table(dados$os), main = "Histograma dos Sist. Operacionais dos Usuários")
```



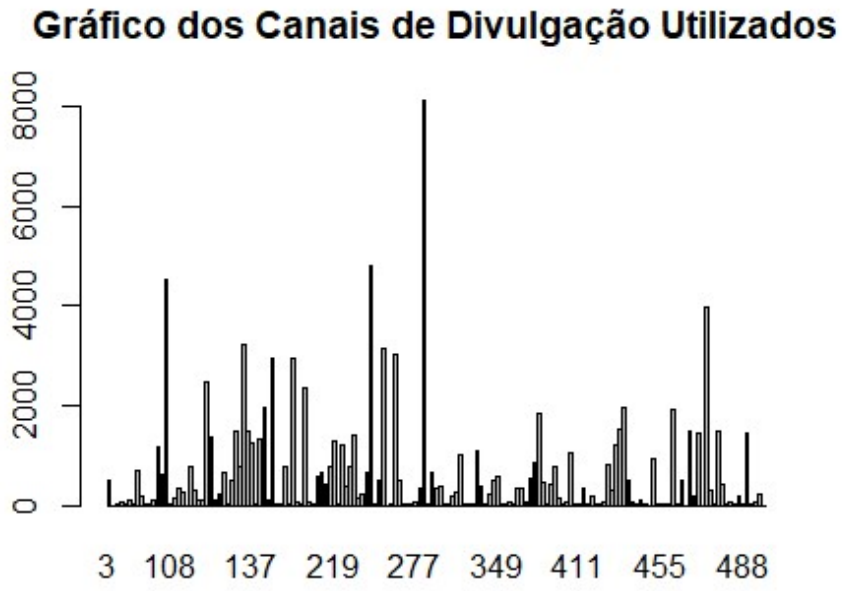

```
hist(table(dados$device), main = "Histograma dos Disp. Celulares dos Usuários")
```



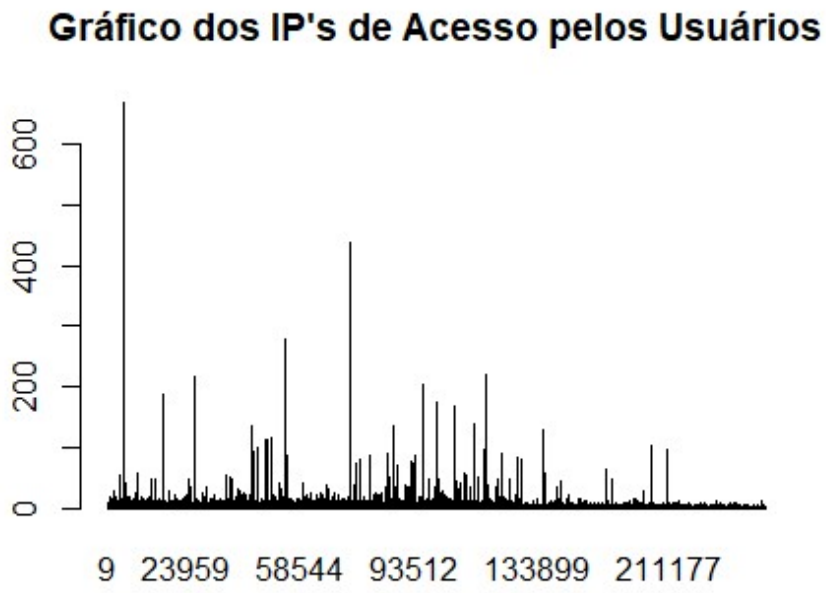
```
hist(table(dados$is_attributed), main = "Hist. dos Total de Aplicativos Baixados")
```




```
# Plot
plot(x = dados$channel, ann = FALSE)
title(main = "Gráfico dos Canais de Divulgação Utilizados")
```



```
# Plot
plot(x = dados$ip, ann = FALSE)
title(main = "Gráfico dos IP's de Acesso pelos Usuários")
```



Distrbuição de dados para Variável Target

Verificando se há necessidade de balanceamento no dataset

```
# Total por condição: 0(dowload não realizado) ou 1 (dowload realizado)
target_count <- table(dados$is_attributed)
target_count
```

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

```
# Proporção em percentual
target_perc <- round(prop.table(target_count) * 100, digits = 1)
target_perc
```

```
##
##    0    1
## 99.8  0.2
```

Observa-se uma desproporção entre as classes a serem apresentadas ao algoritmo (99,8% para a classe 0, e 0,2% para a classe 1), situação que comprometerá o resultado, possivelmente gerando "overffiting" no modelo preditivo.

5. Modelagem Preditiva Sem o Balanceamento do Dataset

Criando um subset

```
df1 <- dados
df1$attributed_time = NULL
df1$click_time = NULL
str(df1)
```

```
## 'data.frame':    100000 obs. of  6 variables:
##  $ ip           : Factor w/ 34857 levels "9","10","19",..: 15221 18449 17664 164
97 11853 16301 2974 21304 28060 23645 ...
##  $ app          : Factor w/ 161 levels "1","2","3","4",..: 12 25 12 13 12 3 1 9
2 3 ...
##  $ device       : Factor w/ 100 levels "0","1","2","4",..: 2 2 2 2 2 2 2 3 2 .
..
##  $ os           : Factor w/ 130 levels "0","1","2","3",..: 14 18 20 14 2 18 18 2
6 23 20 ...
##  $ channel      : Factor w/ 161 levels "3","4","5","13",..: 160 68 53 147 46 21
35 127 101 35 ...
##  $ is_attributed: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Criando os dados de treino e teste

```
# Particionando os dados em treino e teste

amostra = sample(1:nrow(df1), size = 0.7 * nrow(df1))

dados_treino <- df1[amostra, ]
dados_teste <- df1[-amostra, ]

# Conferindo a proporção
table(df1$is_attributed)
```

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

```
prop.table(table(df1$is_attributed))
```

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

```
# Dimensões dos dados
dim(dados_treino)
```

```
## [1] 70000     6
```

```
dim(dados_teste)
```

```
## [1] 30000     6
```

```
# Criando os labels para os dados de treino e teste
dados_treino_label <- df1[amostra, 6]
dados_teste_label <- df1[-amostra, 6]

# Tamanho dos dados
length(dados_treino_label)
```

```
## [1] 70000
```

```
length(dados_teste_label)
```

```
## [1] 30000
```

```
# Criando o modelo preditivo
## Modelo Preditivo com o Algoritmo k-Nearest Neighbour Classification
modelo_knn_v1 <- knn(train = dados_treino,
                     test = dados_teste,
                     cl = dados_treino_label,
                     k = 21)



summary(modelo_knn_v1)
```

```
##     0     1
## 30000     0
```

Embora o modelo "modelo_knn_v1" apresente uma alta taxa de acertos em relação aos dados de teste apresentados, o aprendizado desbalanceado faz com que o modelo torne-se tendencioso em relação a variável target em maior proporção. Nesse caso recomenda-se efetuar o balanceamento dos dados, apresentando ao novo modelo proporções equivalentes das classes "0" e "1".

6. Modelagem Preditiva após Balanceamento do Dataset

Balanceamento do dataset

```
#Carregando o pacote ROSE
library(ROSE)
```

```
## Warning: package 'ROSE' was built under R version 4.1.1
```

```
## Loaded ROSE 0.0-4
```

```
# Efetuando o balanceamento dos dados
# Dados de treino
```

```
rose_treino <- ROSE(is_attributed ~ ., data = dados_treino, seed = 1)$data

# Dimensões
dim(rose_treino)

## [1] 70000     6

prop.table(table(rose_treino$is_attributed))

##
##         0         1
## 0.4988429 0.5011571

# Dados de teste
rose_teste <- ROSE(is_attributed ~ ., data = dados_teste, seed = 1)$data

# Dimensões
dim(rose_teste)

## [1] 30000     6

prop.table(table(rose_teste$is_attributed))

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

Criando os labels para os dados de treino e teste

```
# Labels para dados de treino e teste
rose_treino_label <- df1[amostra, 6]
rose_teste_label <- df1[-amostra, 6]

# Tamanho
length(rose_treino_label)

## [1] 70000

length(rose_teste_label)

## [1] 30000
```

Criando o modelo preditivo

Modelo Preditivo com Algoritmo k-Nearest Neighbour Classification

```
# Construindo um modelo de classificação
modelo_knn_v2 <- knn(train = rose_treino,
                     test = rose_teste,
                     cl = rose_treino_label,
                     k = 21)
```

```
# Resultados obtidos
summary(modelo_knn_v2)
```

```
##     0     1
## 30000     0
```

Ao modelo "modelo_knn_v2" apresentamos no processo aprox. 50% de cada uma das classes da variável target, obtendo um alto índice de acertos.

7. Avaliação dos Modelos Preditivos

Criando uma tabela cruzada dos dados previstos x dados atuais

```
library(gmodels)
# Tabela Cruzada para o Modelo Preditivo 1
CrossTable(x = dados_teste_label, y = modelo_knn_v1, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  30000
##
##
##                    | modelo_knn_v1
## dados_teste_label |         0 | Row Total |
## ------------------|-----------|-----------|
##                 0 |     29935 |     29935 |
##                   |     0.998 |           |
## ------------------|-----------|-----------|
##                 1 |        65 |        65 |
##                   |     0.002 |           |
## ------------------|-----------|-----------|
##      Column Total |     30000 |     30000 |
## ------------------|-----------|-----------|
##
##
```

```
# Tabela Cruzada para o Modelo Preditivo 2
CrossTable(x = rose_teste_label, y = modelo_knn_v2, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
```

```
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  30000
##
##
##                   | modelo_knn_v2
## rose_teste_label |         0 | Row Total |
## -----------------|-----------|-----------|
##                0 |     29935 |     29935 |
##                  |     0.998 |           |
## -----------------|-----------|-----------|
##                1 |        65 |        65 |
##                  |     0.002 |           |
## -----------------|-----------|-----------|
##     Column Total |     30000 |     30000 |
## -----------------|-----------|-----------|
##
##
```

Observa-se que em ambos os casos a Acurácia dos modelos são acima de 90%, porém obtemos com o segundo modelo a capacidade de generalização que será importante ao apresentarmos novos dados para a realização de previsões.