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

Victor Ferreira de Paula

06/06/2020 a 19/06/2020

Considerações gerais

Este relatório documenta o processo de criação de um algoritmo para *Detecção de Fraudes no Tráfego de Cliques em Propagandas de Aplicações Mobile*. Este projeto é parte do Curso *Big Data Analytics com R e Microsoft Azure Machine Learning*, da *Data Science Academy*.

O objetivo deste projeto é prever se um usuário vai realizar o download de um app após clicar na publicidade.

Conforme recomendado, foram utilizados os datasets presentes no kaggle.

Configurando o diretório de trabalho

O diretório de trabalho foi configurado conforme abaixo. Porém, para execução em outro local, deve-se realizar a alteração.

```
setwd("D:/FCD/bigDataRAzure/Cap20-Feedback/Projeto-01")
getwd()
```

Dicionário do dataset

Cada linha do dataset contém dados do histórico de cliques, seguido pelas características:

- *ip:* endereço IP do clique;
- *app:* ID do aplicativo para marketing;
- *device:* ID do tipo de dispositivo do celular do usuário (por exemplo, iphone 6 plus, iphone 7, huawei mate 7 etc);
- *os:* ID da versão do telefone móvel do usuário;
- *channel:* ID do canal do editor de anúncios para celular
- *click_time:* registro de data e hora do clique (UTC);
- *attributed_time:* se o usuário baixar o aplicativo para depois de clicar em um anúncio, este é o horário do download do aplicativo;
- *is_attributed:* a variável a ser prevista, indicando que o aplicativo foi baixado;

Etapa 1: Carregando os dados

Pacotes necessários:

```
library(tidyverse)
```

```
## -- Attaching packages ------------ tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts --------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

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

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(data.table)
```

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

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

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

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
dados <- fread("dados.csv", header = T, stringsAsFactors = F)

glimpse(dados)
```

```
## Observations: 100,000
## Variables: 8
## $ ip              <int> 87540, 105560, 101424, 94584, 68413, 93663, 17059, ...
## $ app             <int> 12, 25, 12, 13, 12, 3, 1, 9, 2, 3, 3, 3, 3, 6, 2, 2...
## $ device          <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, ...
## $ os              <int> 13, 17, 19, 13, 1, 17, 17, 25, 22, 19, 22, 13, 22, ...
## $ channel         <int> 497, 259, 212, 477, 178, 115, 135, 442, 364, 135, 4...
## $ click_time      <chr> "2017-11-07 09:30:38", "2017-11-07 13:40:27", "2017...
## $ attributed_time <chr> "", "", "", "", "", "", "", "", "", "", "", "", "",...
## $ is_attributed   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

Etapa 2: Pré-processamento

Transformando as variáveis *click_time* e *attributed_time* em data usando o pacote *lubridate*:

```
dados$click_time <- ymd_hms(dados$click_time)
dados$attributed_time <- ymd_hms(dados$attributed_time)
```

Para classificar em qual período do dia um clique ou download é realizado, cria-se a função conforme abaixo. Para isso, temos os seguintes períodos do dia:

- 1 = Madrugada: 00:00:00 a 05:59:59 horas
- 2 = Manhã: 06:00:00 a 11:59:59 horas
- 3 = Tarde: 12:00:00 a 17:59:59 horas
- 4 = Noite: 18:00:00: a 23:59:59 horas
- 0 = Não foi realizado download

```
# Retorna o perído do dia de alguma horário
day_period <- function(x) {

    if ( !is.na(x) ){
        hora = hour(x)

        if ( hora <= 5) { periodo = 1 }
        if ( 6 <= hora & hora <= 11 ) { periodo = 2 }
        if ( 12 <= hora & hora <= 17 ) { periodo = 3 }
        if ( 18 <= hora & hora <= 23 ) { periodo = 4 }
    } else {
        periodo = 0
    }

    return(periodo)
}
```

Testando a função:

```
datas <- c("2017-11-18 07:01:20",
           "2005-12-23 15:05:34",
           "2020-01-07 05:20:25",
           "2015-11-11 20:43:59",
           NA)

sapply(datas, day_period)
```

```
## 2017-11-18 07:01:20 2005-12-23 15:05:34 2020-01-07 05:20:25 2015-11-11 20:43:59
##                   2                   3                   1                   4
##                <NA>
##                   0
```

Criando uma coluna com o período do dia para as variáveis *click_time* e *attributed_time*:

```
dados$period_click_time <-
    as.factor(sapply(dados$click_time, day_period))

dados$period_attributed_time <-
    as.factor(sapply(dados$attributed_time,day_period))
```

Classificando a variável alvo como fator:

```
dados$is_attributed <- factor(dados$is_attributed)
```

Sumário do pré-processamento:

```
glimpse(dados)
```

```
## Observations: 100,000
## Variables: 10
```

```
## $ ip                    <int> 87540, 105560, 101424, 94584, 68413, 93663, ...
## $ app                   <int> 12, 25, 12, 13, 12, 3, 1, 9, 2, 3, 3, 3, 3, ...
## $ device                <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,...
## $ os                    <int> 13, 17, 19, 13, 1, 17, 17, 25, 22, 19, 22, 1...
## $ channel               <int> 497, 259, 212, 477, 178, 115, 135, 442, 364,...
## $ click_time            <dttm> 2017-11-07 09:30:38, 2017-11-07 13:40:27, 2...
## $ attributed_time       <dttm> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ is_attributed         <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ period_click_time     <fct> 2, 3, 4, 1, 2, 1, 1, 2, 2, 3, 2, 1, 2, 3, 1,...
## $ period_attributed_time <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

Etapa 3: Análise Exploratória

Verificando a proporção, observamos que 99,77% dos cliques não originaram em download do app. Isso ilustra um desbalanceamento muito elevado dos dados.

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

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

Tendência Central - Cliques por IP:

```
dados %>%
    count(ip) %>%
    select(n) -> clicks

summary(clicks$n)
```

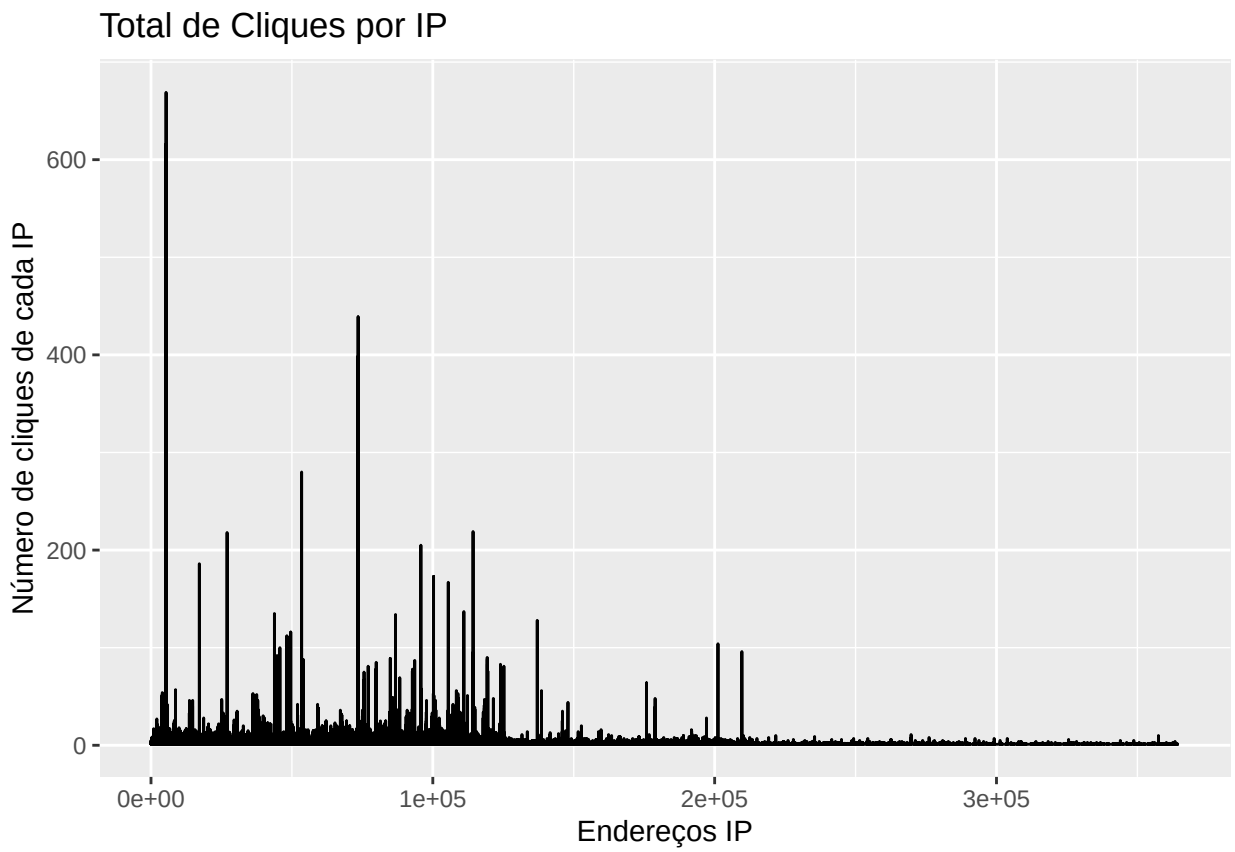
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   2.869   3.000 669.000
```

```
# Alguns percentis:
quantile(clicks$n, c(0.8, 0.9, 0.95, 0.99))
```

```
## 80% 90% 95% 99%
##   3   6   8  18
```

Histograma - Cliques por IP:

```
# "Histograma" dos cliques por IP:
dados %>%
    count(ip) %>%
    ggplot(mapping = aes(x = as.numeric(ip), y = n) ) +
        geom_line() +
        ggtitle("Total de Cliques por IP") +
        ylab("Número de cliques de cada IP") +
        xlab("Endereços IP")
```



Fração de cliques convertidos em download:

```
dados %>%
    select(ip, is_attributed) %>%
    filter(is_attributed == 1) %>%
    count(ip) %>%
    summarise(total = sum(n)) -> dwd_clicks

as.numeric(dwd_clicks) / sum(clicks$n)
```

```
## [1] 0.00227
```

Podemos considerar valores *outliers* como aqueles que estão 3σ (sendo σ o desvio-padrão) distantes do valor médio. Nesse caso, vamos filtrar os números de ip com mais de $\mu + 3\sigma \approx 27$ cliques.

```
mu <- mean(clicks$n)
sigma <- sd(clicks$n)

mu + 3*sigma
```

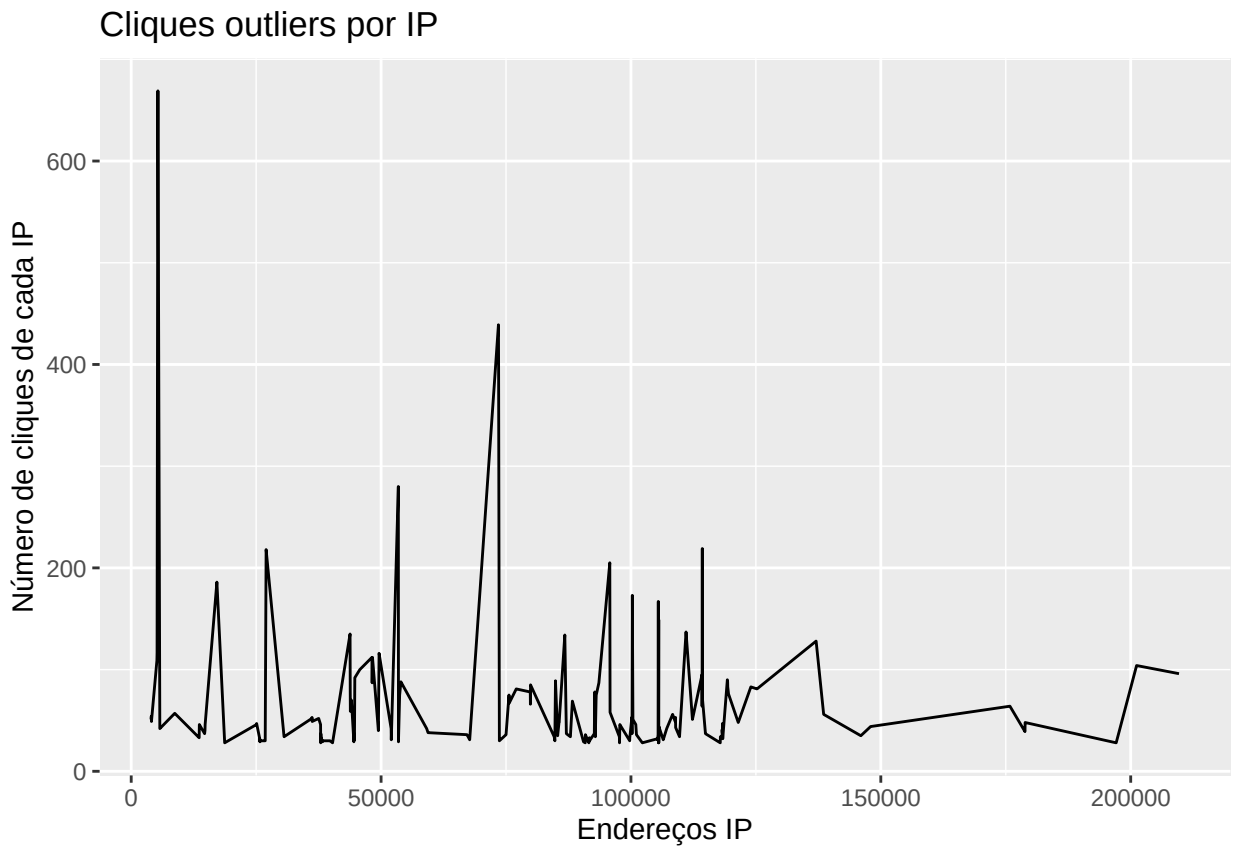
```
## [1] 26.99841
```

```
# Contagem dos outliers:
clicks %>%
    filter( n > round(mu + 3*sigma)) %>%
    summarise(total = sum(n)) -> clicks_out

# Proporcão de cliques outliers:
as.numeric(clicks_out) / sum(clicks$n)
```

```
## [1] 0.11923
```

```
# Histograma dos cliques outliers por IP:
dados %>%
    count(ip) %>%
    filter(n > round(mu + 3*sigma)) %>%
    ggplot(mapping = aes(x = as.numeric(ip), y = n)) +
        geom_line() +
        ggtitle("Cliques outliers por IP") +
        ylab("Número de cliques de cada IP") +
        xlab("Endereços IP")
```



Vamos verificar quanto dos cliques outliers realizaram o download do app.

```
dados %>%
    select(ip, is_attributed) %>%
    filter(is_attributed == 1) %>%
    count(ip) %>%
    summarise(total = sum(n)) -> clicks_out_download
```

Fração dos cliques outliers que fizeram download:

```
as.numeric(clicks_out_download) / clicks_out
```

```
##       total
## 1 0.01903883
```

Fração dos cliques outliers com mais de 200 cliques:

```
dados %>%
    count(ip) %>%
    filter(n >= 200) %>%
    summarise(total = sum(n)) -> clicks_out_200

as.numeric(clicks_out_200) / clicks_out
```

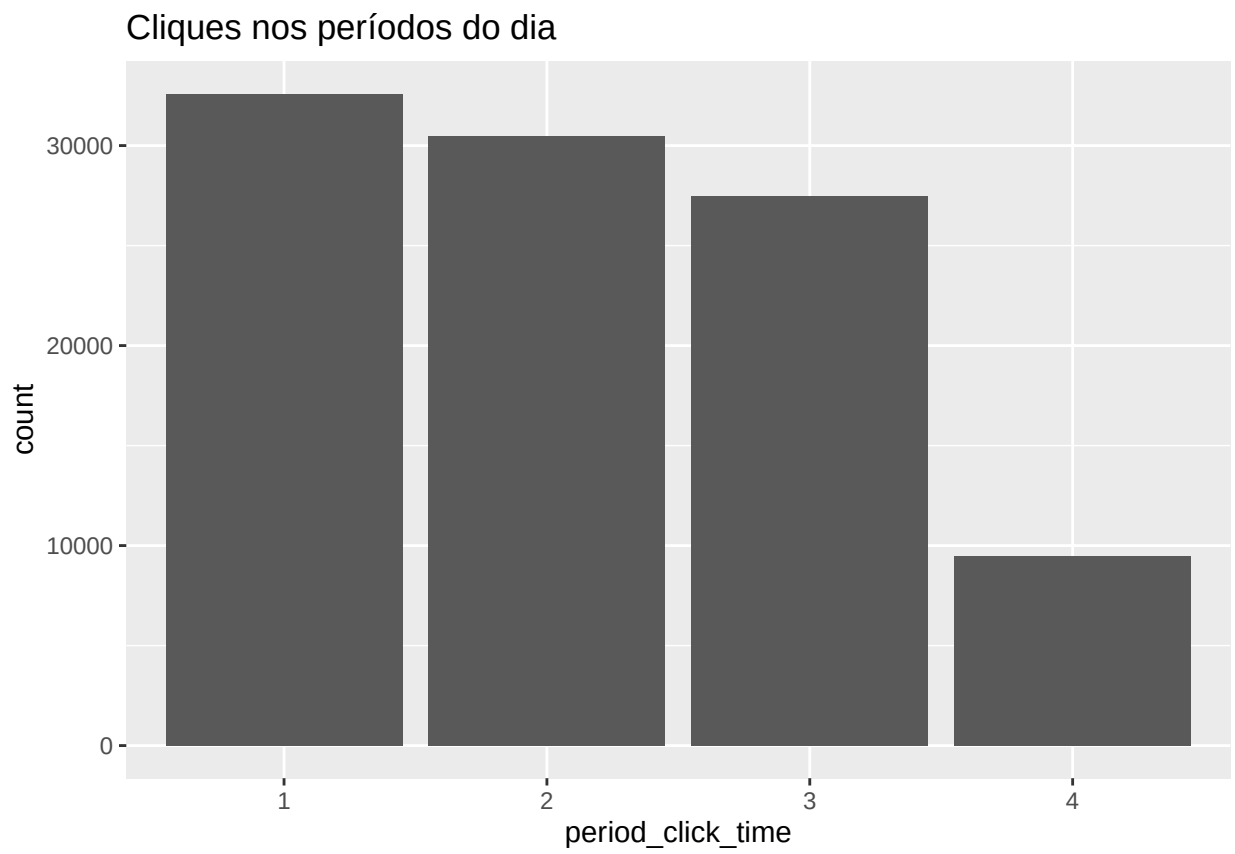
```
##       total
## 1 0.2553887
```

```
as.numeric(clicks_out_200) / sum(clicks$n)
```

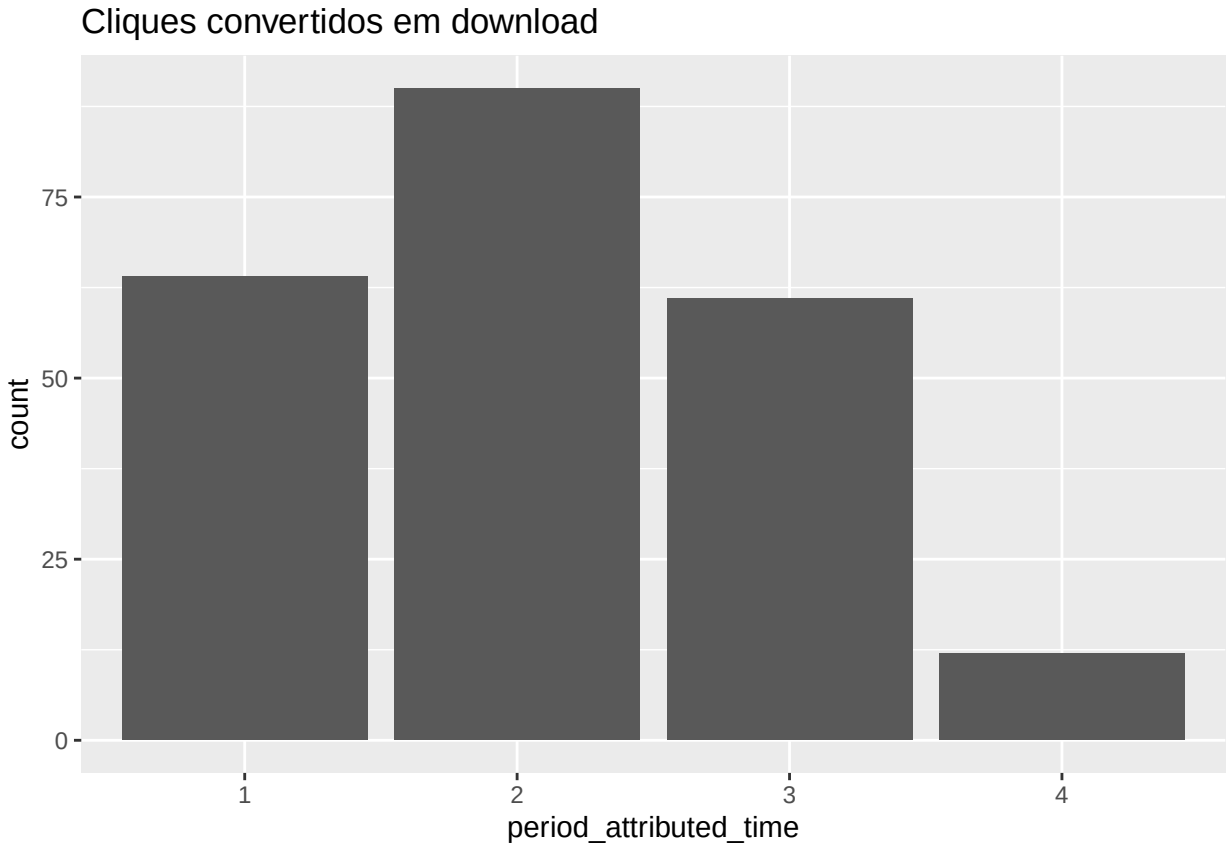
```
## [1] 0.03045
```

Cliques por período do dia:

```
ggplot(data = dados) +
    geom_bar(aes(x = period_click_time)) +
    ggtitle("Cliques nos períodos do dia")
```




```
ggplot(data = subset(dados, period_attributed_time != 0)) +
    geom_bar(aes(x = period_attributed_time)) +
    ggtitle("Cliques convertidos em download")
```



Cliques convertidos de manhã:

```
dados %>%
    select(ip, period_attributed_time) %>%
    filter(period_attributed_time == 2) %>%
    count(ip) %>%
    summarise(total = sum(n)) -> clicks_dwd_morning

as.numeric(clicks_dwd_morning) / as.numeric(dwd_clicks)
```

```
## [1] 0.3964758
```

Para os cliques convertidos, diferença entre o tempo de download e o tempo do clique:

```
diff_tempos <- data.frame(ip = dados$ip,
               diff=dados$attributed_time - dados$click_time)

# Eliminando os valores NA:
diff_tempos <- subset( diff_tempos, !is.na(diff_tempos$diff) )

# Colocando em minutos, para facilitar a compreensão:
diff_tempos$diff <- diff_tempos$diff / 60

# Medidas de Tendência Central (em minutos):
summary(as.numeric(diff_tempos$diff))
```

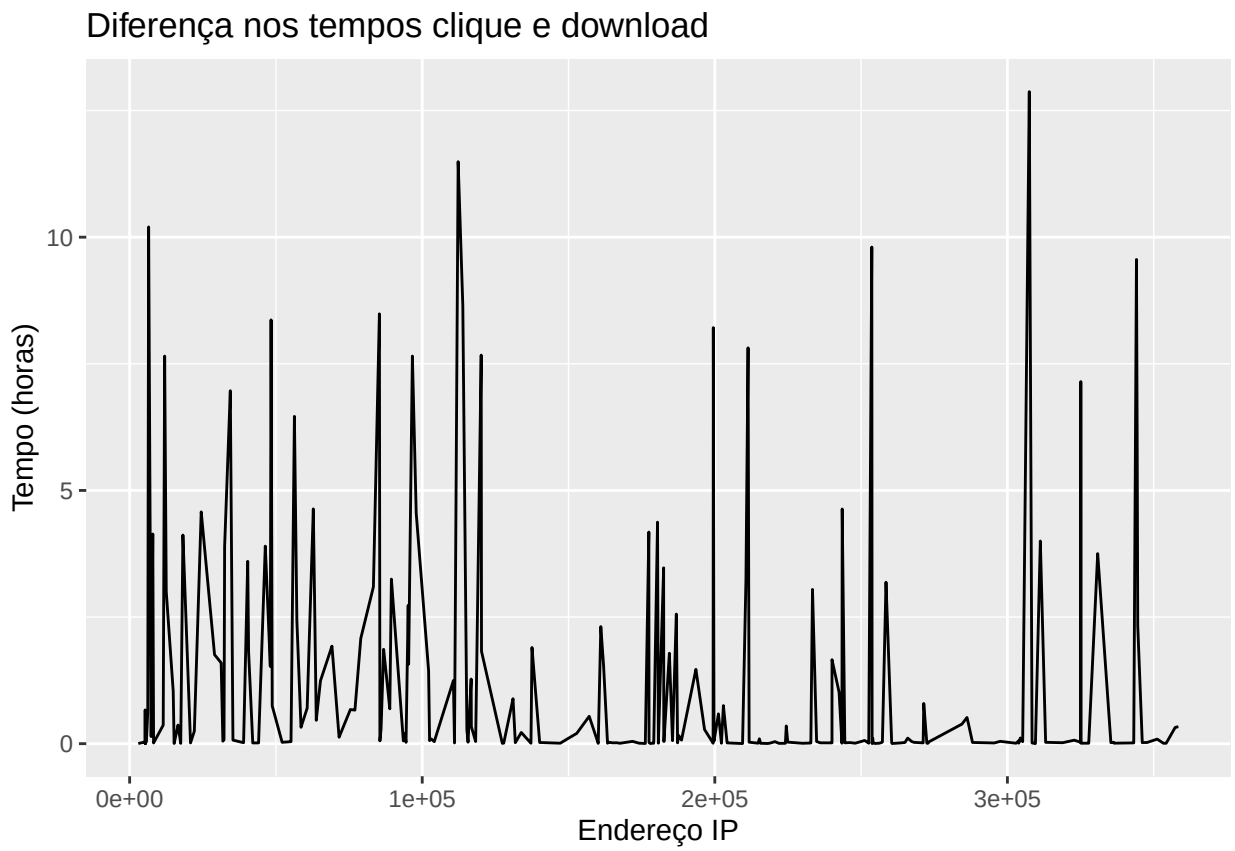
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0333   0.8750   3.3000  74.9929  81.4583 772.3500
```

```r
# Alguns percentis (em minutos):
quantile(diff_tempos$diff, c(0.8, 0.9, 0.95, 0.99))
```

```
## Time differences in secs
##      80%      90%      95%      99%
## 111.2967 249.2867 459.1567 605.9163
```

```r
# "Histograma" usando geom_line():
ggplot(data =  diff_tempos, aes(y = diff / 60, x = as.numeric(ip))) +
    geom_line() +
    ggtitle("Diferença nos tempos clique e download") +
    xlab("Endereço IP") +
    ylab("Tempo (horas)")
```

```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



Para o modelo, não usaremos as variáveis de data *click_time* e *attributed_time*, mas sim as variáveis *period_click_time* e *period_attributed_time*.

```r
dados$click_time <- NULL
dados$attributed_time <- NULL
```

Verificando a associação entre as variáveis categóricas:

```r
chisq.test(x = dados$is_attributed, y = dados$ip)
```

```
## Warning in chisq.test(x = dados$is_attributed, y = dados$ip): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  dados$is_attributed and dados$ip
## X-squared = 74845, df = 34856, p-value < 2.2e-16
```

```
chisq.test(x = dados$is_attributed, y = dados$app)
```

```
## Warning in chisq.test(x = dados$is_attributed, y = dados$app): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  dados$is_attributed and dados$app
## X-squared = 20302, df = 160, p-value < 2.2e-16
```

```
chisq.test(x = dados$is_attributed, y = dados$device)
```

```
## Warning in chisq.test(x = dados$is_attributed, y = dados$device): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  dados$is_attributed and dados$device
## X-squared = 9557.7, df = 99, p-value < 2.2e-16
```

```
chisq.test(x = dados$is_attributed, y = dados$os)
```

```
## Warning in chisq.test(x = dados$is_attributed, y = dados$os): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  dados$is_attributed and dados$os
## X-squared = 6431.3, df = 129, p-value < 2.2e-16
```

```
chisq.test(x = dados$is_attributed, y = dados$channel)
```

```
## Warning in chisq.test(x = dados$is_attributed, y = dados$channel): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  dados$is_attributed and dados$channel
## X-squared = 17146, df = 160, p-value < 2.2e-16
```

```
chisq.test(x = dados$is_attributed, y = dados$period_click_time)
```

```
##
##  Pearson's Chi-squared test
##
## data:  dados$is_attributed and dados$period_click_time
## X-squared = 2.3682, df = 3, p-value = 0.4996
```

```
chisq.test(x = dados$is_attributed, y = dados$period_attributed_time)
```

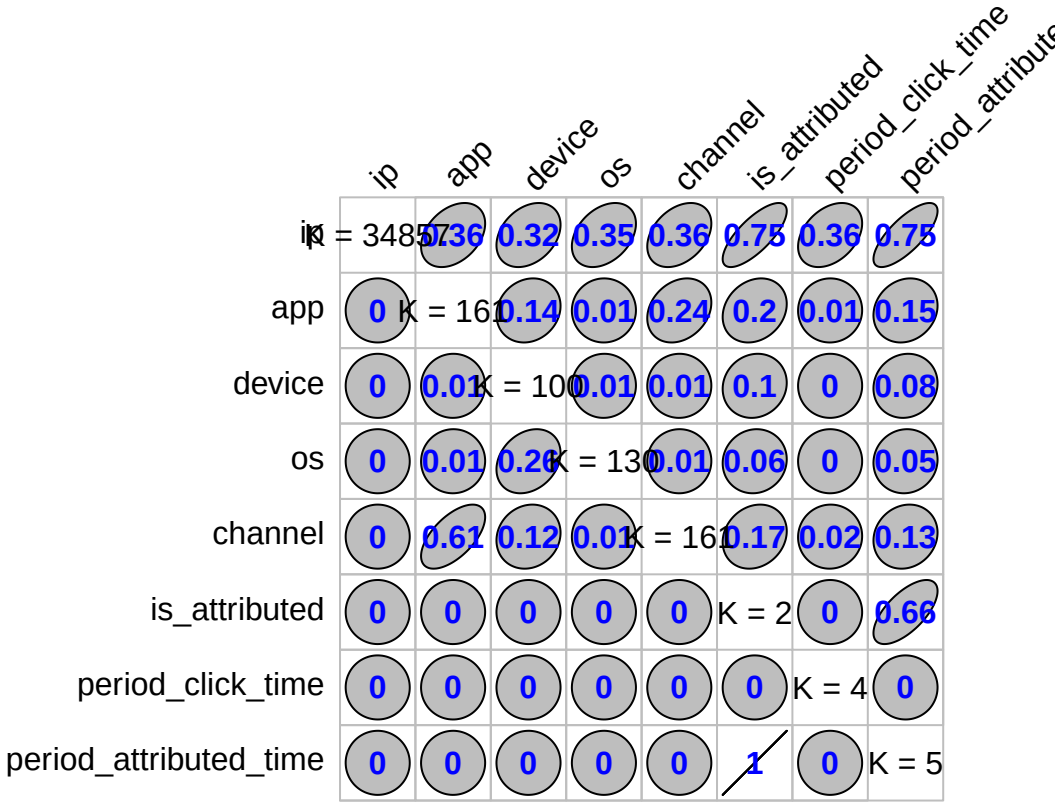
```
## Warning in chisq.test(x = dados$is_attributed, y =
## dados$period_attributed_time): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  dados$is_attributed and dados$period_attributed_time
## X-squared = 1e+05, df = 4, p-value < 2.2e-16
```

Verificando a força da associação das variáveis com o teste de tau de Goodman e Kruskal:

```
library(GoodmanKruskal)

GK_matrix <- GKtauDataframe(dados)
plot(GK_matrix, corrColors = "blue")
```



Algumas conclusões da Análise Exploratória dos dados:

- Dados altamente desbalancedos, podendo acarretar em *overfitting*;
- Cada endereço de IP gera entre 2 e 3 cliques;
- 80% dos endereços de IP geram 3 cliques;
- 0,2% dos cliques são efetivamente convertidos em download;
- Cerca de 12% dos endereços de IP dão mais de 27 cliques, sendo suspeitos de fraude;
- Dos IPs com mais de 27 cliques, 2% realizam o download;
- Dos IPs com mais de 27 cliques, 25% dão mais que 200 cliques de um total de 100,000 cliques;

- Os IPs com mais que 200 cliques representam 3% dos total de cliques;
- Cerca de 10% dos cliques é realizado no período da madrugada;
- Dos cliques convertidos em download, 40% no período da manhã;
- A média entre clique e download do app é de 75 minutos, aproximadamente;
- A conversão de download está ligada ao enderço de IP, mais do que as outras características;

Etapa 4: Criação do modelo

Selecionando dados de treino e teste:

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
trainIndex <- createDataPartition(dados$is_attributed,
                                  p = 0.7,
                                  list = FALSE)

treino <- dados[ trainIndex, ]
teste  <- dados[-trainIndex, ]
```

Observamos que os dados de treino estão desbalanceados:

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

```
##
##           0           1
## 0.997728604 0.002271396
```

Balancemento dos dados de treino e teste usando ROSE:

```
library(ROSE)
```

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

```
treino_rose <- ROSE(is_attributed ~ . , data = treino)$data
```

Agora os dados de treino estão balanceados:

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

```
##
##         0         1
## 0.5019357 0.4980643
```

Modelo: K-Nearest Neighbor Classification do pacote class.

```
library(class)

modelo_v1 <- knn(train = treino_rose,
                 test = teste,
```

```
                cl = treino_rose[, 6],
                k = 6)
```

Etapa 5: Avaliação do modelo

Modelo 1:

```
previsao <- modelo_v1

# Percentual de previsões corretas com dataset de teste
mean(previsao == teste$is_attributed)
```

```
## [1] 0.8418947
```

```
# Confusion Matrix
library(gmodels)
CrossTable(x = previsao, y = teste$is_attributed, chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  29999
##
##
##              | teste$is_attributed
##     previsao |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |     25223 |        35 |     25258 |
##              |     0.020 |     8.649 |           |
##              |     0.999 |     0.001 |     0.842 |
##              |     0.843 |     0.515 |           |
##              |     0.841 |     0.001 |           |
## -------------|-----------|-----------|-----------|
##            1 |      4708 |        33 |      4741 |
##              |     0.105 |    46.081 |           |
##              |     0.993 |     0.007 |     0.158 |
##              |     0.157 |     0.485 |           |
##              |     0.157 |     0.001 |           |
## -------------|-----------|-----------|-----------|
## Column Total |     29931 |        68 |     29999 |
##              |     0.998 |     0.002 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Etapa 6: Otimização

Cabe observar que todas as variáveis do dataset *dados* são do tipo fator.

```
dados$ip <- factor(dados$ip)
dados$app <- factor(dados$app)
dados$device <- factor(dados$device)
dados$os <- factor(dados$os)
dados$channel <- factor(dados$channel)
dados$is_attributed <- factor(dados$is_attributed)
dados$period_click_time <- factor(dados$period_click_time)
dados$period_attributed_time <- factor(dados$period_attributed_time)
```

Vamos estudar como o modelo knn se comporta quando todas as variáveis são classificadas corretamente.

```
# Dados de treino e teste
trainIndex <- createDataPartition(dados$is_attributed,
                                  p = 0.7,
                                  list = FALSE)

treino <- dados[ trainIndex, ]
teste  <- dados[-trainIndex, ]

# Balanceamento com ROSE
treino_rose <- ROSE(is_attributed ~ . , data = treino)$data

# Criação do modelo
modelo_v2 <- knn(train = treino_rose,
                 test = teste,
                 cl = treino_rose[, 6],
                 k = 6)

previsao <- modelo_v2

# Percentual de previsões corretas com dataset de teste
mean(previsao == teste$is_attributed)
```

```
## [1] 0.9760992
```

```
# Confusion Matrix
CrossTable(x = previsao, y = teste$is_attributed, chisq = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  29999
##
```

```
##
##              | teste$is_attributed
##     previsao |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |     29273 |        59 |     29332 |
##              |     0.002 |     0.843 |           |
##              |     0.998 |     0.002 |     0.978 |
##              |     0.978 |     0.868 |           |
##              |     0.976 |     0.002 |           |
## -------------|-----------|-----------|-----------|
##            1 |       658 |         9 |       667 |
##              |     0.084 |    37.086 |           |
##              |     0.987 |     0.013 |     0.022 |
##              |     0.022 |     0.132 |           |
##              |     0.022 |     0.000 |           |
## -------------|-----------|-----------|-----------|
## Column Total |     29931 |        68 |     29999 |
##              |     0.998 |     0.002 |           |
## -------------|-----------|-----------|-----------|
##
##
```

A simples reclassificação das variáveis no dataset de treino acarretou num aumento expressivo na performance do modelo.

Conclusão

O Modelo Knn criado conseguiu uma acurácia maior que 90% para classificação dos dados apresentados. Porém, um estudo mais detalhado dos dados pode ser feito se precisarmos buscar uma performance melhor, como a divisão entre outros períodos do dia nos cliques, ao invés dos realizados aqui, ou seja, subdividir manhã, tarde, noite e madrugada em mais períodos).

Outro ponto que pode ser realizado para generalização do modelo é um *feature selection* visando escolher as características mais representativas da conversão de cliques em download.