TalkingData AdTracking - Projeto do Curso 1 da Formação Cientista de Dados DSA

Carlos Paiva

03 de Outubro de 2020

Detectar cliques fraudulentos em anúncios de app mobile

Projeto da Formação Cientista de Dados da Data Science Academy.

Este projeto serve para prever se um clique em anúncio é fraudulendo ou não.

Entende-se como fraudulento quando clica no anúncio, mas não faz o download (is_attributed == 0)

Dataset usado: https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data?select=train_sample.csv

Carregando o dataset

```
# Coletando os dados
dataset <- read.csv("train_sample.csv")
```

Feature Selection e Feature Engineering

```
dataset.v1 <- dataset
dataset.v1$attributed_time <- NULL

# Agora quero dividir a click_time em duas colunas
# click_date e click_time
# click_date vai armazenar apenas data
# click_hour vai armazenar apenas hora

# Extraindo datas e convertendo para POSIXct
dates <- as.POSIXct(dataset.v1$click_time)

# Pegando datas
dataset.v1$click_date <- format(dates, format = "%Y/%m/%d")

# Pegando o dia da semana com base na data
dataset.v1$click_weekday <- weekdays(as.Date(dataset.v1$click_date))

# Pegando horas (apenas hora, estou ignorando os minutos e segundos)
dataset.v1$click_hour <- as.numeric(format(dates, format = "%H"))
```

```r
# Vou remover o campo click_time pois já tenho os dados que preciso
#   e não faz sentido manter dados duplicados
dataset.v1$click_time <- NULL

# Tranformando variáveis em fator
dataset.v1$click_date <- as.factor(dataset.v1$click_date)

# Função para agrupar as horas de acordo com MEUS parâmetros de partes do dia
# Observação: Esses horários foram definidos por MIM, não quer dizer que precisem ser a regra geral
# Manhã - De 5 até 12
# Tarde - De 12 até 19
# Noite - De 19 até 5
group_day_part <- function(x){
  if(x>5 && x<=12){
    return("Manha")
  }else if(x>12 && x<=19){
    return("Tarde")
  }else{
    return("Noite")
  }
}

# Armazenado as partes do dia em uma variável nova
# Preciso usar o unlist() senão não vou conseguir converter para fator, pois após o lapply
#   a variável fica do tipo list
dataset.v1$day_part <- unlist(lapply(dataset.v1$click_hour, group_day_part))
dataset.v1$day_part <- as.factor(dataset.v1$day_part)
dataset.v1$click_weekday <- as.factor(dataset.v1$click_weekday)
dataset.v1$is_attributed <- as.factor(dataset.v1$is_attributed)

# Decidi remover as horas pois já tenho a informação que queria (parte do dia)
dataset.v1$click_hour <- NULL

head(dataset.v1)
```

```
##       ip app device os channel is_attributed click_date click_weekday day_part
## 1  87540  12      1 13     497             0 2017/11/07   terça-feira    Manha
## 2 105560  25      1 17     259             0 2017/11/07   terça-feira    Tarde
## 3 101424  12      1 19     212             0 2017/11/07   terça-feira    Tarde
## 4  94584  13      1 13     477             0 2017/11/07   terça-feira    Noite
## 5  68413  12      1  1     178             0 2017/11/09  quinta-feira    Manha
## 6  93663   3      1 17     115             0 2017/11/09  quinta-feira    Noite
```

```r
str(dataset.v1)
```

```
## '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: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ click_date   : Factor w/ 4 levels "2017/11/06","2017/11/07",..: 2 2 2 2 4 4 4 2 3 3 ...
##  $ click_weekday: Factor w/ 4 levels "quarta-feira",..: 4 4 4 4 2 2 2 4 1 1 ...
##  $ day_part     : Factor w/ 3 levels "Manha","Noite",..: 1 3 3 2 1 2 2 1 1 1 ...
```

Análise Exploratória

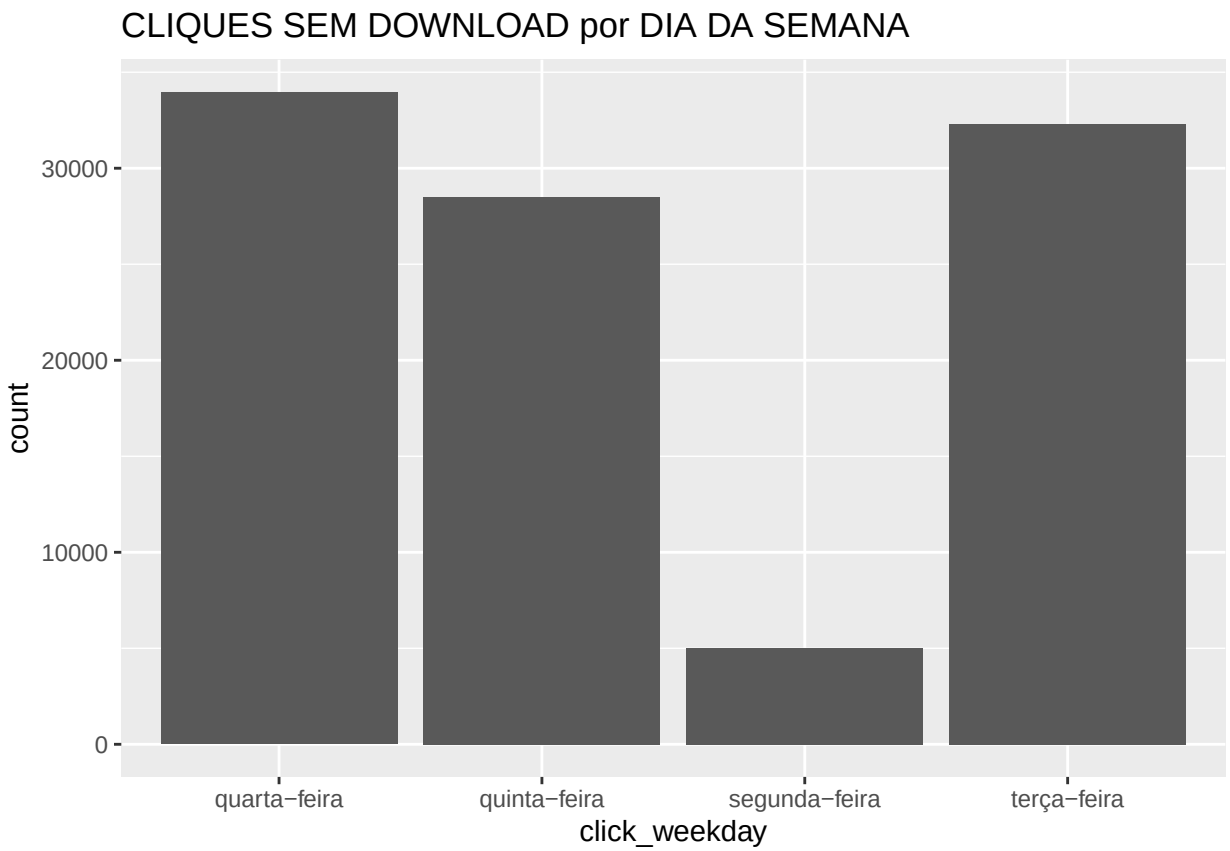
```
table(dataset.v1$is_attributed)
```

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

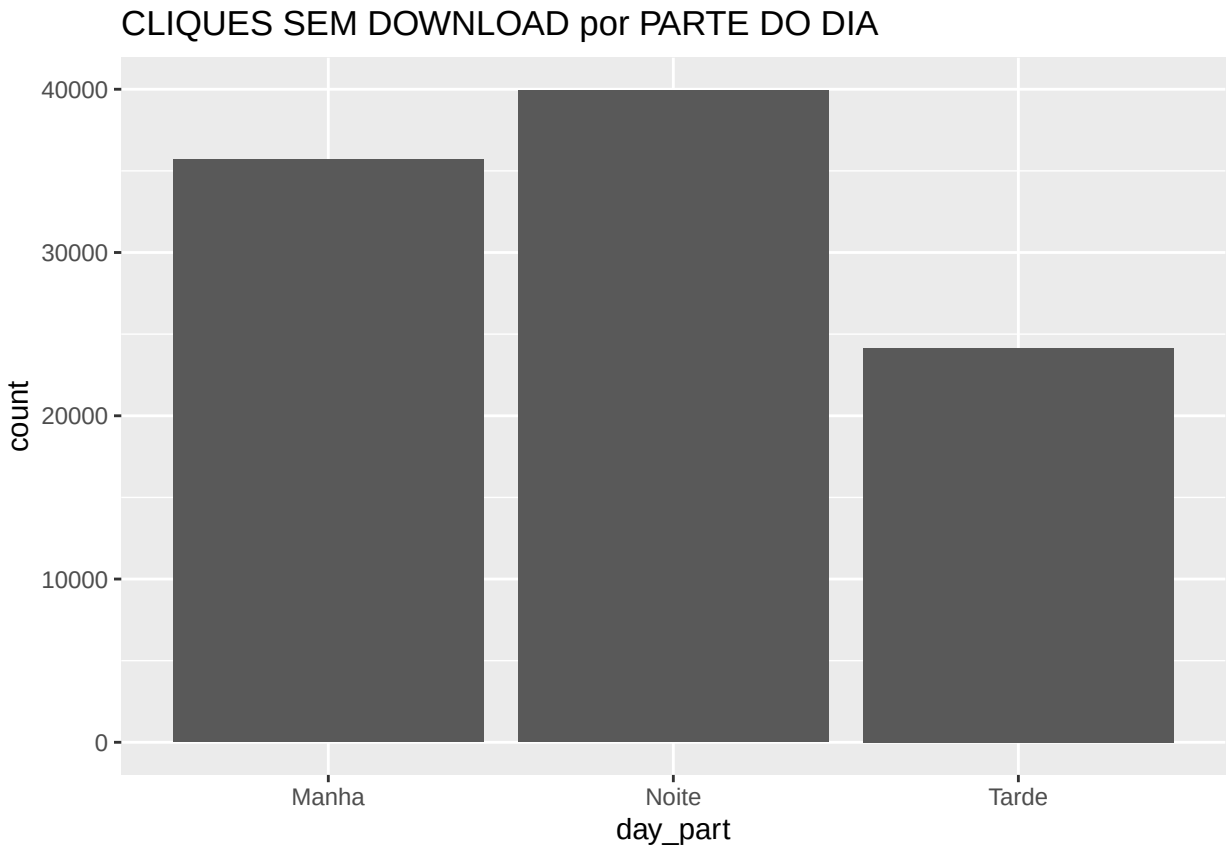
```
# Como já era de se esperar, existem mais downloads NÃO efetuados

table(dataset.v1$click_weekday)
```

```
##
##  quarta-feira  quinta-feira segunda-feira   terça-feira
##         34035         28561          5011         32393
```

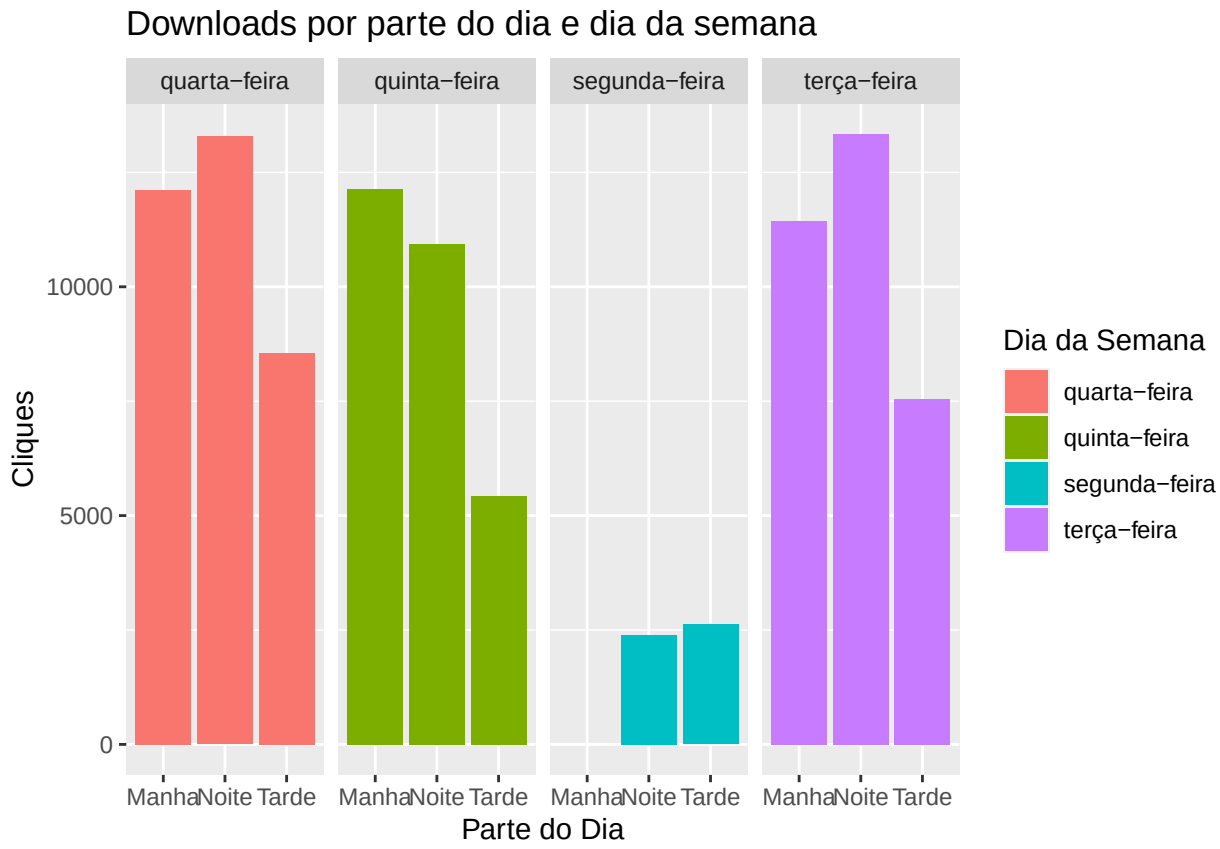
```
# Já nessa outra tabela, observa-se que existem menos ocorrências na segunda-feira
# E maior numero de ocorrências na quarta
# Indicando que com base nesses dados, e nas datas analisadas, o dia da semana que
#   os anúncios são mais clicados são na quarta-feira

library(ggplot2)
ggplot(dataset.v1[dataset.v1$is_attributed == 0,], aes(x = click_weekday)) +
  geom_bar() +
  ggtitle("CLIQUES SEM DOWNLOAD por DIA DA SEMANA")
```




```
ggplot(dataset.v1[dataset.v1$is_attributed == 0,], aes(x = day_part)) +
  geom_bar() +
ggtitle("CLIQUES SEM DOWNLOAD por PARTE DO DIA")
```




```
ggplot(dataset.v1[dataset.v1$is_attributed == 0,], aes(x = day_part, fill = click_weekday)) +
  labs(fill = "Dia da Semana") +
  geom_bar() +
  facet_grid(. ~ click_weekday) +
  ylab("Cliques") +
  xlab("Parte do Dia") +
  ggtitle("Downloads por parte do dia e dia da semana")
```




```
# Como podemos ver, na terça e quarta a noite ocorrem a maioria dos cliques
# Sendo esse o período com maior chance de ocorrer alguma fraude (clique sem download)
# Essa informação pode ajudar a empresa a se preparar para esses dias e períodos, além de permitir
#   que outras estratégias sejam adotadas para melhorar os números de cliques nos outros dias da semana
```

Balanceamento

```
table(dataset.v1$is_attributed)
```

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

```
# Os dados estão desbalanceados
# Balanceando os dados através de undersampling
#   (diminuir os dados com maior quantidade com base nos dados de menor quantidade)

# Separando as duas categorias de is_attributed (0 e 1)
dataset.v1.0 <- dataset.v1[dataset.v1$is_attributed == 0,]
dataset.v1.1 <- dataset.v1[dataset.v1$is_attributed == 1,]

# Escolhendo os dados de forma randômica
dataset.v1.0 <- dataset.v1.0[sample(1:nrow(dataset.v1.1)),]
```

```r
# Unindo os dois datasets
dataset.v2 <- merge(dataset.v1.0, dataset.v1.1, all = T)

head(dataset.v2)
```

```
##     ip app device os channel is_attributed click_date click_weekday day_part
## 1 2600  12      1 18     265             0 2017/11/07   terça-feira    Tarde
## 2 2948  45      1  2     419             1 2017/11/09  quinta-feira    Tarde
## 3 3488   3      1 22     115             0 2017/11/07   terça-feira    Noite
## 4 4019  12      1 19     265             0 2017/11/09  quinta-feira    Noite
## 5 4865  19      0 24     213             1 2017/11/09  quinta-feira    Noite
## 6 5281  35      1 13      21             1 2017/11/09  quinta-feira    Noite
```

```r
str(dataset.v2)
```

```
## 'data.frame':    454 obs. of  9 variables:
##  $ ip           : int  2600 2948 3488 4019 4865 5281 5314 5314 5314 5314 ...
##  $ app          : int  12 45 3 12 19 35 2 10 18 28 ...
##  $ device       : int  1 1 1 1 0 1 1 1 1 1 ...
##  $ os           : int  18 2 22 19 24 13 2 1 19 19 ...
##  $ channel      : int  265 419 115 265 213 21 477 113 107 135 ...
##  $ is_attributed: Factor w/ 2 levels "0","1": 1 2 1 1 2 2 1 2 2 1 ...
##  $ click_date   : Factor w/ 4 levels "2017/11/06","2017/11/07",..: 2 4 2 4 4 4 4 2 3 2 ...
##  $ click_weekday: Factor w/ 4 levels "quarta-feira",..: 4 2 4 2 2 2 2 4 1 4 ...
##  $ day_part     : Factor w/ 3 levels "Manha","Noite",..: 3 3 2 2 2 2 1 2 1 3 ...
```

```r
table(dataset.v2$is_attributed)
```

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

```r
# São poucos dados mas agora não vai tender mais pra um dos lados
```

Normalização

```r
# As variáveis estão com escala diferentes

# Função para alterar a escala
scale.features <- function(df, variables){
  for (variable in variables){
    df[[variable]] <- scale(df[[variable]], center=T, scale=T)
  }
  return(df)
}

# Normalizando as variáveis
numeric.vars <- c('ip', 'app', 'device', 'os', 'channel')
dataset.v2 <- scale.features(dataset.v2, numeric.vars)
```

```r
# Separando dados em treino (70%) e teste (30%)
train_data <- dataset.v2[1:round(nrow(dataset.v2) * 0.7),]
test_data <- dataset.v2[(round(nrow(dataset.v2) * 0.7)+1):(nrow(dataset.v2)),]
```

Treinamento do Modelo

```r
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
```

```r
# Primeiramente estou criando o modelo usando todas as variáveis
model.rf.v1 <- randomForest(is_attributed ~ .,
                            data = train_data,
                            ntree = 100,
                            nodesize = 10)

# Imprimindo resultado do treinamento v1
print(model.rf.v1)
```

```
##
## Call:
##  randomForest(formula = is_attributed ~ ., data = train_data,      ntree = 100, nodesize = 10)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 9.43%
## Confusion matrix:
##     0  1 class.error
## 0 200  6  0.02912621
## 1  24 88  0.21428571
```

Avaliação do modelo

```r
# Previsão usando dados de teste e gravando o resultado
predict.rf <- data.frame(observado = test_data$is_attributed,
                            previsto = predict(model.rf.v1, newdata = test_data))

# Visualizando resultado
head(predict.rf)
```

```
##     observado previsto
## 319         1        1
## 320         1        1
## 321         1        1
## 322         1        1
## 323         1        0
## 324         1        1
```

```
# Criando a confusion matrix
library(caret)
```

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

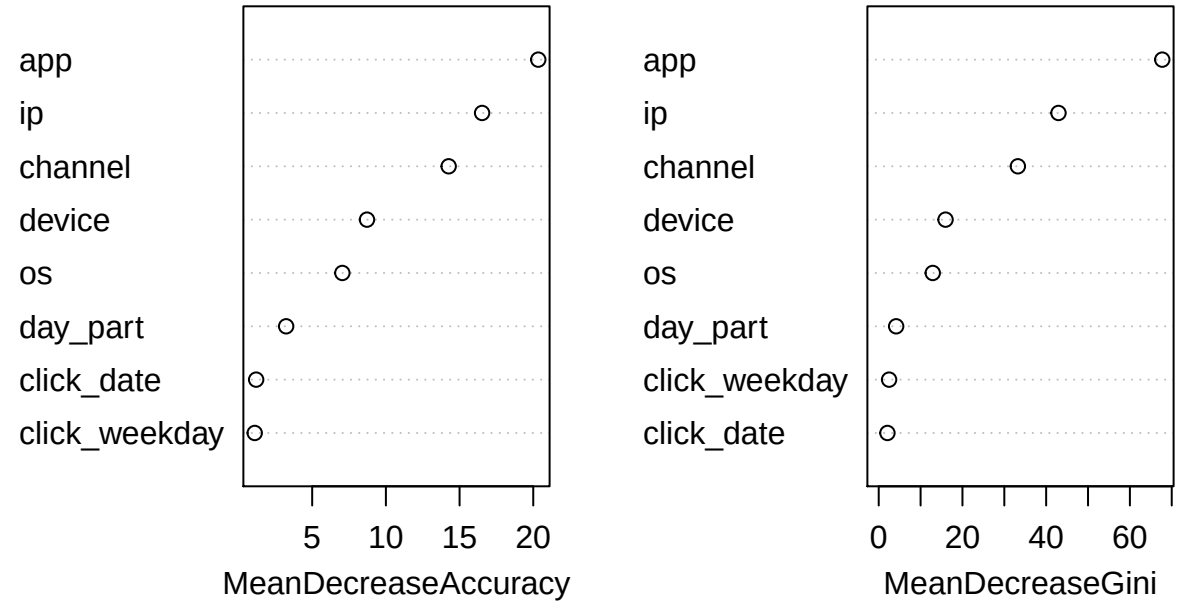
```
confusionMatrix(predict.rf$observado, predict.rf$previsto)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20  1
##          1 31 84
##
##                Accuracy : 0.7647
##                  95% CI : (0.6844, 0.8332)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : 0.0003657
##
##                   Kappa : 0.4311
##
##  Mcnemar's Test P-Value : 2.951e-07
##
##             Sensitivity : 0.3922
##             Specificity : 0.9882
##          Pos Pred Value : 0.9524
##          Neg Pred Value : 0.7304
##              Prevalence : 0.3750
##          Detection Rate : 0.1471
##    Detection Prevalence : 0.1544
##       Balanced Accuracy : 0.6902
##
##        'Positive' Class : 0
##
```

Otimizando o Modelo

```
# Alterando variáveis usadas pelo modelo
# Selecionando variáveis com randomForest
variables.rf <- randomForest(is_attributed ~ .,
                             data = dataset.v2,
                             ntree = 100,
                             nodesize = 10,
```

```
                            importance = TRUE)
# Visualizando resultado
varImpPlot(variables.rf)
```




```
# Recriando modelo com as variáveis mais importantes de acordo com o resultado acima
model.rf.v2 <- randomForest(is_attributed ~ app
                            + ip
                            + channel
                            + device
                            + os,
                            data = train_data,
                            ntree = 100,
                            nodesize = 10)

# Imprimindo resultado do treinamento v2
print(model.rf.v2)
```

```
##
## Call:
##  randomForest(formula = is_attributed ~ app + ip + channel + device +      os, data = train_data, nt
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 10.38%
```

```
## Confusion matrix:
##     0  1 class.error
## 0 199  7  0.03398058
## 1  26 86  0.23214286
```

Avaliação do Modelo v2

```
predict.rf.v2 <- data.frame(observado = test_data$is_attributed,
                            previsto = predict(model.rf.v2, newdata = test_data))

# Visualizando resultado
View(predict.rf.v2)

# Criando uma confusion matrix
library(caret)
confusionMatrix(predict.rf.v2$observado, predict.rf.v2$previsto)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  21   0
##          1  12 103
##
##                Accuracy : 0.9118
##                  95% CI : (0.8509, 0.9536)
##     No Information Rate : 0.7574
##     P-Value [Acc > NIR] : 3.207e-06
##
##                   Kappa : 0.7261
##
##  Mcnemar's Test P-Value : 0.001496
##
##             Sensitivity : 0.6364
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.8957
##              Prevalence : 0.2426
##          Detection Rate : 0.1544
##    Detection Prevalence : 0.1544
##       Balanced Accuracy : 0.8182
##
##        'Positive' Class : 0
##
```

Como observado na confusion matrix acima, obtivemos uma melhora na acurácia, apenas alterando as variáveis usadas para o treinamento.