

Projeto - Detecção de Fraude

Raphael Serra de Oliveira

6/30/2020

Este trabalho tem o objetivo de prever se um usuário fará o download de um aplicativo depois de clicar em um anúncio de aplicativo para celular.

A maior plataforma independente de serviço de big data da China, cobre mais de 70% dos dispositivos móveis ativos em todo o país. Eles processam 3 bilhões de cliques por dia, dos quais 90% são potencialmente fraudulentos. Sua abordagem atual para evitar a fraude de cliques para desenvolvedores de aplicativos é medir a jornada do clique de um usuário em seu portfólio e sinalizar endereços IP que produzem muitos cliques, mas nunca acabam instalando aplicativos.

Etapa 1 - Carregando Bibliotecas e Objetos do Script Auxiliar

```
suppressMessages(library(readr))
suppressMessages(library(knitr))
suppressMessages(library(dplyr))
suppressMessages(library(ggplot2))
suppressMessages(library(caret))
suppressMessages(library(lubridate))
suppressMessages(library(gridExtra))
suppressMessages(library(randomForest))
suppressMessages(library(DMwR))
suppressMessages(library(ROCR))
suppressMessages(library(e1071))
library(readr)
library(knitr)
library(dplyr)
library(ggplot2)
library(caret)
library(lubridate)
library(gridExtra)
library(randomForest)
library(DMwR)
library(ROCR)
library(e1071)
source("src/Tools.R")
```

Etapa 2 - Carregando dados de treino

```
# Carregando dados de treino
train_sample <- read_csv("train_sample.csv")
```

```
## Parsed with column specification:
## cols(
##   ip = col_double(),
##   app = col_double(),
##   device = col_double(),
##   os = col_double(),
##   channel = col_double(),
##   click_time = col_datetime(format = ""),
##   attributed_time = col_datetime(format = ""),
##   is_attributed = col_double()
## )

# Algumas informações úteis sobre o dataset
dim(train_sample)

## [1] 100000      8

str(train_sample)

## tibble [100,000 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##   $ ip          : num [1:100000] 87540 105560 101424 94584 68413 ...
##   $ app         : num [1:100000] 12 25 12 13 12 3 1 9 2 3 ...
##   $ device      : num [1:100000] 1 1 1 1 1 1 1 1 2 1 ...
##   $ os          : num [1:100000] 13 17 19 13 1 17 17 25 22 19 ...
##   $ channel     : num [1:100000] 497 259 212 477 178 115 135 442 364 135 ...
##   $ click_time  : POSIXct[1:100000], format: "2017-11-07 09:30:38" "2017-11-07 13:40:27" ...
##   $ attributed_time: POSIXct[1:100000], format: NA NA ...
##   $ is_attributed : num [1:100000] 0 0 0 0 0 0 0 0 0 0 ...
##   - attr(*, "spec")=
##     .. cols(
##     ..   ip = col_double(),
##     ..   app = col_double(),
##     ..   device = col_double(),
##     ..   os = col_double(),
##     ..   channel = col_double(),
##     ..   click_time = col_datetime(format = ""),
##     ..   attributed_time = col_datetime(format = ""),
##     ..   is_attributed = col_double()
##     .. )
```

Etapa 3 - Feature Selection

```
train_sample <- train_sample %>%
  mutate(wday = as.factor(weekdays(click_time, abbreviate=T))) %>%
  mutate(hour = hour(click_time)) %>%
  select(-c(click_time)) %>%
  add_count(ip, wday, hour) %>% rename("nip_day_h" = n) %>%
  add_count(ip, hour, channel) %>% rename("nip_h_chan" = n) %>%
  add_count(ip, hour, os) %>% rename("nip_h_osr" = n) %>%
  add_count(ip, hour, app) %>% rename("nip_h_app" = n) %>%
  add_count(ip, hour, device) %>% rename("nip_h_dev" = n) %>%
  select(-c(ip, attributed_time))
```

nip_day_h = número de cliques de um mesmo IP, no mesmo dia e na mesma hora

nip_h_chan = número de cliques de um mesmo IP, no mesma hora e do mesmo

```
# canal de anúncio
# nip_h_osr = número de cliques de um mesmo IP, na mesma hora e de um mesmo OS
# nip_h_app = número de cliques de um mesmo IP, na mesma hora e no mesmo APP
# nip_h_dev = número de cliques de um mesmo IP, na mesma hora e no mesmo
# dispositivo
```

Etapa 4 - Análise Exploratória

```
# Número de valores únicos por variável
unique_values <- as.data.frame(lapply(train_sample, function(x)length(unique(x))))
unique_values
```

```
##   app device  os channel is_attributed wday hour nip_day_h nip_h_chan nip_h_osr
## 1 161    100 130    161             2   4   24         27           9        16
##   nip_h_app nip_h_dev
## 1         12        39
```

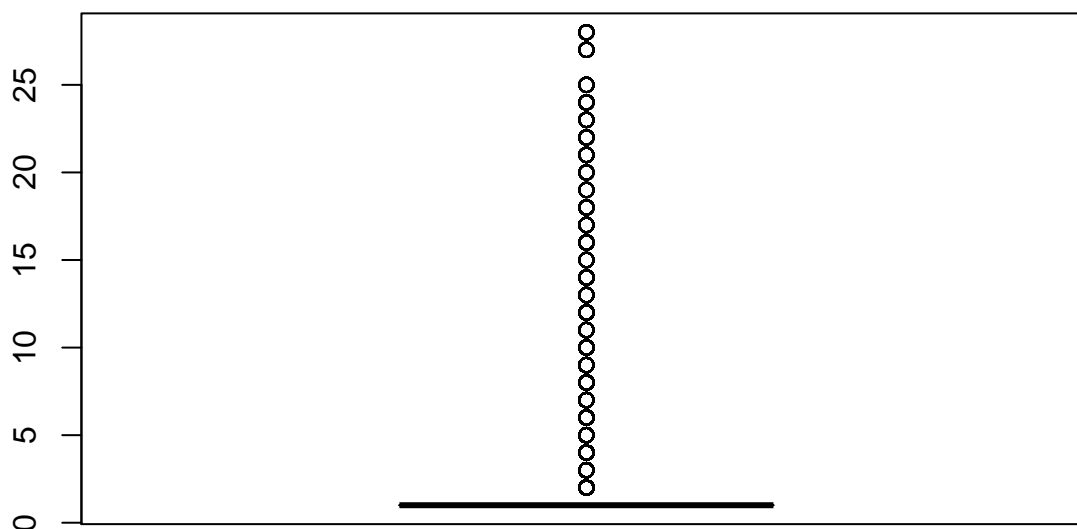
```
# Verificando valores missing
sapply(train_sample, function(x) sum(is.na(x)))
```

```
##           app           device           os           channel is_attributed
##           0             0           0           0             0
##          wday           hour    nip_day_h    nip_h_chan    nip_h_osr
##           0             0           0           0             0
##   nip_h_app    nip_h_dev
##           0             0
```

```
# Visualizando a distribuição e os outliers das variáveis criadas durante
# o processo de feature selection
```

```
# Os boxplots não nos traz tanta informação visual nesse caso,
# pois os valores de primeiro quartil, mediana e terceiro quartil
# estão muito próximo. Porém é interessante notar a presença
# dos outliers
```

```
boxplot(train_sample$nip_day_h)
```



```
# Podemos ver que a média de clicks por ip em um mesmo dia e hora é aproximadamente 1.
# Portanto os outliers podem indicar a ação de bots.
```

```
summary(train_sample$nip_day_h)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.000   1.000   1.000   1.493   1.000  28.000
```

```
sd(train_sample$nip_day_h)
```

```
## [1] 2.020593
```

```
# O mesmo padrão se repete para as outras variáveis
```

```
summary(train_sample$nip_h_chan)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.000   1.000   1.000   1.054   1.000  10.000
```

```
sd(train_sample$nip_h_chan)
```

```
## [1] 0.3332162
```

```
summary(train_sample$nip_h_osr)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.000   1.000   1.000   1.154   1.000  16.000
```

```
sd(train_sample$nip_h_osr)
```

```
## [1] 0.8607566
```

```
summary(train_sample$nip_h_app)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.000   1.000   1.000   1.145   1.000  12.000
```

```
sd(train_sample$nip_h_app)
```

```
## [1] 0.7217136
```

```
summary(train_sample$nip_h_dev)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.000   1.000   1.000   2.054   2.000  59.000
```

```
sd(train_sample$nip_h_dev)
```

```
## [1] 4.370674
```

```
# Plot de frequência das variáveis APP, Device, OS, Channel, Hora e Dia da semana
```

```
# filtrando o dataset para dados onde não houve download
```

```
h1 <- train_sample %>% group_by(app) %>% filter(is_attributed==FALSE) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(app)) %>% head(15) %>%
  ggplot(aes(x = reorder(app, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top Apps") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) + labs(x = "APP")

h2 <- train_sample %>% group_by(device) %>% filter(is_attributed==FALSE) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(device = as.character(device)) %>% head(15) %>%
  ggplot(aes(x = reorder(device, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top Devices") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) +
  labs(x = "Device")
```

```

h3 <- train_sample %>% group_by(os) %>% filter(is_attributed==FALSE) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(os)) %>% head(15) %>%
  ggplot(aes(x = reorder(os, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top OS") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) + labs(x = "OS")

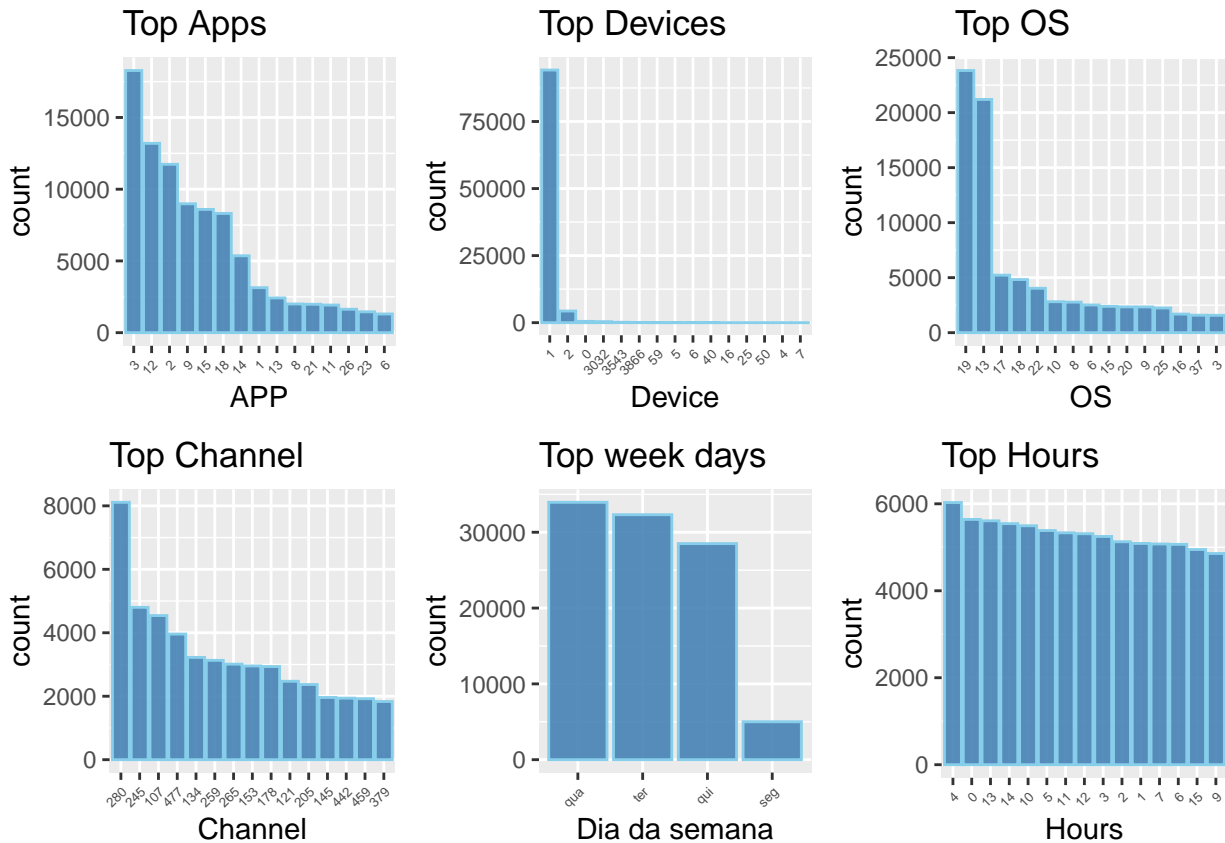
h4 <- train_sample %>% group_by(channel) %>% filter(is_attributed==FALSE) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(channel)) %>% head(15) %>%
  ggplot(aes(x = reorder(channel, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top Channel") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) +
  labs(x = "Channel")

h5 <- train_sample %>% group_by(wday) %>% filter(is_attributed==FALSE) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(wday)) %>% head(15) %>%
  ggplot(aes(x = reorder(wday, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top week days") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) +
  labs(x = "Dia da semana")

h6 <- train_sample %>% group_by(hour) %>% filter(is_attributed==FALSE) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(hour)) %>% head(15) %>%
  ggplot(aes(x = reorder(hour, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top Hours") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) +
  labs(x = "Hours")

grid.arrange(h1, h2, h3, h4, h5, h6, nrow = 2)

```



*# Plot de frequência das variáveis APP, Device, OS, Channel, Hora e dia da semana filtrando o dataset
para dados onde houve o download*

```
h7 <- train_sample %>% filter(is_attributed==TRUE) %>% group_by(app) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(app)) %>% head(15) %>%
  ggplot(aes(x = reorder(app, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top Apps") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) + labs(x = "APP")

h8 <- train_sample %>% filter(is_attributed==TRUE) %>% group_by(device) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(device)) %>% head(15) %>%
  ggplot(aes(x = reorder(device, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top Devices") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) +
  labs(x = "Device")

h9 <- train_sample %>% filter(is_attributed==TRUE) %>% group_by(os) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(os)) %>% head(15) %>%
  ggplot(aes(x = reorder(os, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top OS") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) + labs(x = "OS")

h10 <- train_sample %>% filter(is_attributed==TRUE) %>% group_by(channel) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(channel)) %>% head(15) %>%
  ggplot(aes(x = reorder(channel, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +
  ggtitle("Top Channel") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) +
```

```

labs(x = "Channel")

h11 <- train_sample %>% filter(is_attributed==TRUE) %>% group_by(wday) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(wday)) %>% head(15) %>%
  ggplot(aes(x = reorder(wday, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +

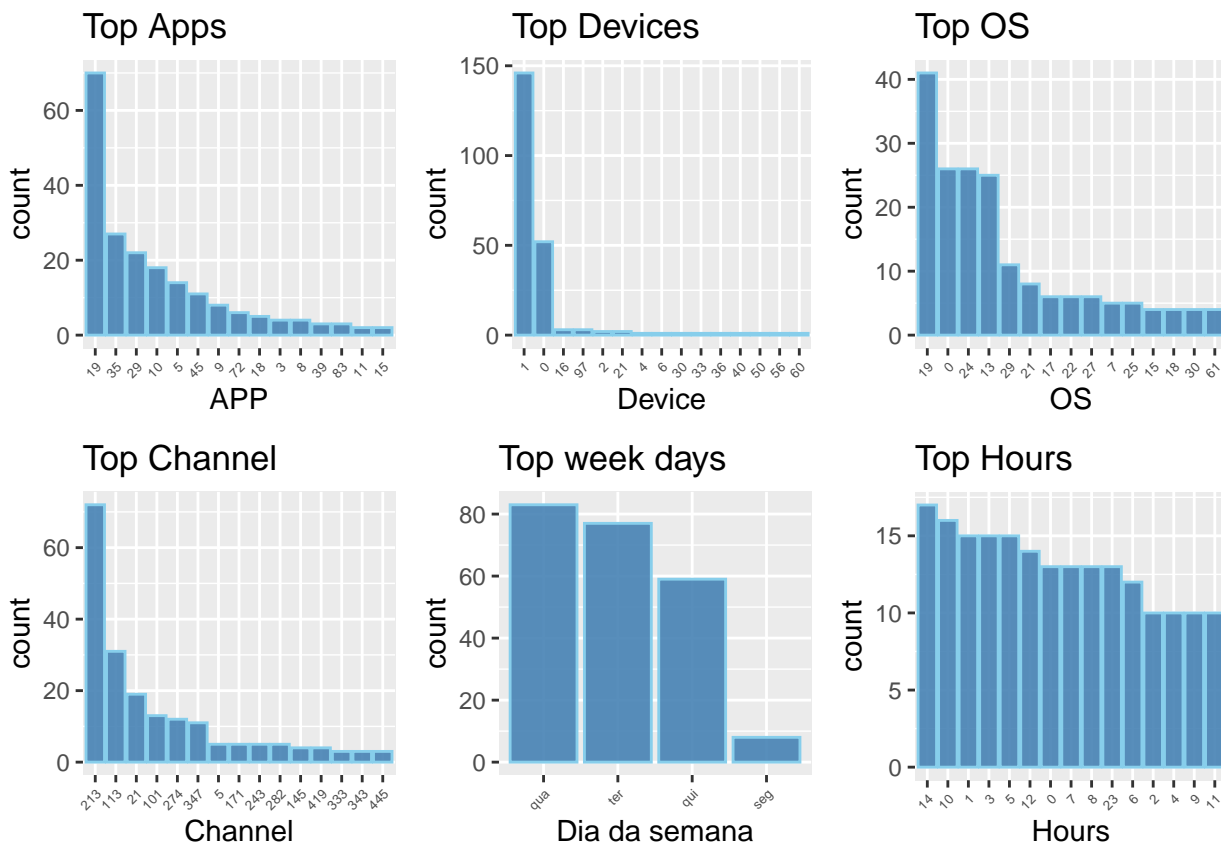
  ggtitle("Top week days") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) +
  labs(x = "Dia da semana")

h12 <- train_sample %>% filter(is_attributed==TRUE) %>% group_by(hour) %>% summarise(count = n()) %>%
  arrange(desc(count)) %>% mutate(app = as.character(hour)) %>% head(15) %>%
  ggplot(aes(x = reorder(hour, -count), y=count)) + geom_bar(stat='identity', color='skyblue',
    fill="steelblue", alpha=0.9) +

  ggtitle("Top Hours") + theme(axis.text.x=element_text(angle=45, hjust=1, size=4.5)) +
  labs(x = "Hours")

grid.arrange(h7, h8, h9, h10, h11, h12, nrow = 2)

```



*# Podemos observar certas diferenças na utilização de aplicativo, dispositivo, hora e outros em
relação aos dados onde tivemos o download e onde não tivemos o download.*

Etapa 5 - Transformando variáveis em fator

```

train_sample <- to.factors(train_sample, factColNames)
str(train_sample)

```

```
## tibble [100,000 x 12] (S3: tbl_df/tbl/data.frame)
## $ app      : num [1:100000] 12 25 12 13 12 3 1 9 2 3 ...
## $ device   : num [1:100000] 1 1 1 1 1 1 1 1 2 1 ...
## $ os       : num [1:100000] 13 17 19 13 1 17 17 25 22 19 ...
## $ channel  : num [1:100000] 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 ...
## $ wday     : Factor w/ 4 levels "qua","qui","seg",...: 4 4 4 4 2 2 2 4 1 1 ...
## $ hour     : Factor w/ 24 levels "0","1","2","3",...: 10 14 19 5 10 2 2 11 10 13 ...
## $ nip_day_h : int [1:100000] 1 4 1 1 1 1 1 1 1 1 ...
## $ nip_h_chan : int [1:100000] 1 1 1 1 1 1 1 1 1 1 ...
## $ nip_h_osr  : int [1:100000] 1 1 1 1 1 1 1 1 1 1 ...
## $ nip_h_app  : int [1:100000] 1 1 1 1 1 1 1 1 1 1 ...
## $ nip_h_dev  : int [1:100000] 1 8 1 1 1 1 2 1 1 1 ...
```

Etapa 6 - Verificando balanceamento do dataset

```
table(train_sample$is_attributed)

##
##      0      1
## 99773  227

prop.table(table(train_sample$is_attributed))

##
##      0      1
## 0.99773 0.00227

# Dataset altamente desbalanceado

# Balanceamento do dataset
balanced_train_sample <- SMOTE(is_attributed ~ .,
                               as.data.frame(train_sample),
                               k = 3,
                               perc.over = 400,
                               perc.under = 150)

table(balanced_train_sample$is_attributed)

##
##      0      1
## 1362 1135

# Podemos ver que o balanceamento funcionou
prop.table(table(balanced_train_sample$is_attributed))

##
##      0      1
## 0.5454545 0.4545455
```

Etapa 7 - Machine Learning

```
# Dividindo o dataset em dados de treino e dados de teste
set.seed(123)
smp_size <- floor(0.70 * nrow(balanced_train_sample))
train_ind <- sample(seq_len(nrow(balanced_train_sample)), size = smp_size)
```



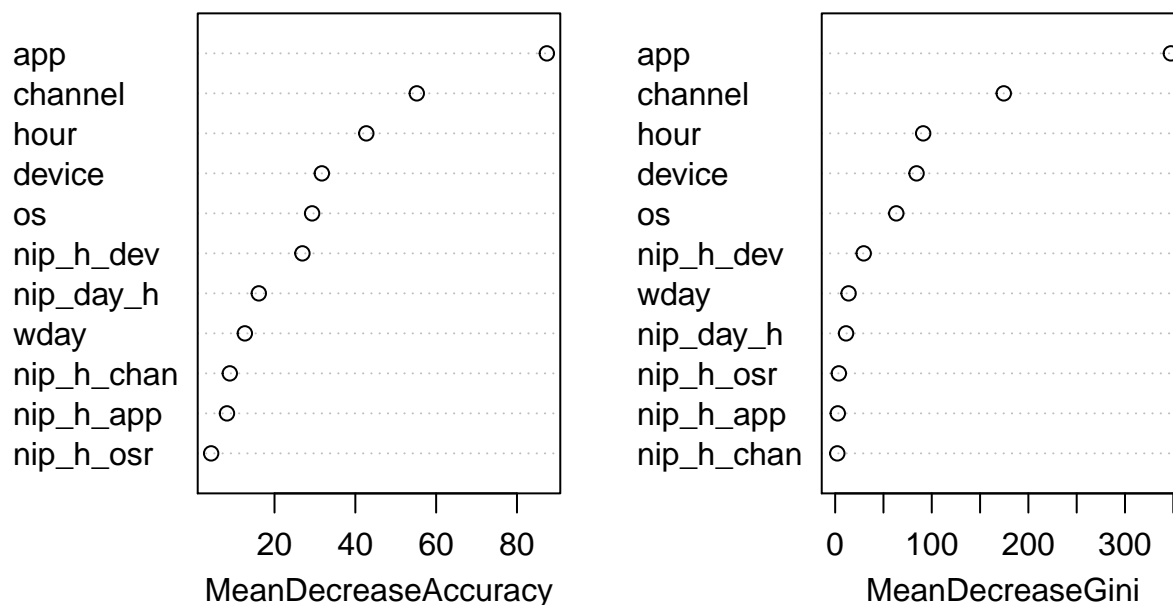
```

train <- balanced_train_sample[train_ind, ]
test <- balanced_train_sample[-train_ind, ]

# Analisando relevância das variáveis para o modelo preditivo
# Aqui utilizo o algoritmo randomforest como ferramenta para averiguar a importância das variáveis
# para o modelo preditivo
modelo_rf1 <- randomForest(is_attributed ~ ., data=train, importance=TRUE)
varImpPlot(modelo_rf1)

```

modelo_rf1



```

# Treinando novamente o modelo com as 7 variáveis de maior relevância
# Foi adotado o método Gini para a escolha das variáveis
modelo_rf1 <- randomForest(is_attributed ~ app +
                           channel +
                           hour +
                           device +
                           os +
                           nip_h_dev +
                           nip_day_h,
                           data=train)

# Análise da performance do modelo
# Criando dataframe com valores observados historicamente e com os valores previstos pelo
# modelo de machine learning
score_model <- data.frame(observado = test$is_attributed,
                          previsto = predictions <- predict(modelo_rf1, test[, -5]))

confusionMatrix(score_model$observado, score_model$previsto)

```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0    1
##           0 385  13
##           1  22 330
##
##           Accuracy : 0.9533
##           95% CI : (0.9357, 0.9673)
##           No Information Rate : 0.5427
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9062
##
## Mcnemar's Test P-Value : 0.1763
##
##           Sensitivity : 0.9459
##           Specificity : 0.9621
##           Pos Pred Value : 0.9673
##           Neg Pred Value : 0.9375
##           Prevalence : 0.5427
##           Detection Rate : 0.5133
##           Detection Prevalence : 0.5307
##           Balanced Accuracy : 0.9540
##
##           'Positive' Class : 0
##
```

```
modelo_rf2 <- randomForest(is_attributed ~ app +
                           channel +
                           hour +
                           device +
                           os +
                           nip_h_dev +
                           nip_day_h,
                           data=train,
                           ntree = 300,
                           nodesize = 3)

score_model2 <- data.frame(observado = test$is_attributed,
                           previsto = predictions <- predict(modelo_rf2, test[, -5]))

confusionMatrix(score_model2$observado, score_model2$previsto)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0    1
##           0 384  14
##           1  19 333
##
##           Accuracy : 0.956
##           95% CI : (0.9388, 0.9695)
##           No Information Rate : 0.5373
##           P-Value [Acc > NIR] : <2e-16
```

```
##
##           Kappa : 0.9116
##
## Mcnemar's Test P-Value : 0.4862
##
##           Sensitivity : 0.9529
##           Specificity : 0.9597
##           Pos Pred Value : 0.9648
##           Neg Pred Value : 0.9460
##           Prevalence : 0.5373
##           Detection Rate : 0.5120
##           Detection Prevalence : 0.5307
##           Balanced Accuracy : 0.9563
##
##           'Positive' Class : 0
##
```

```
# Testando outro algoritmo de classificação SVM
```

```
# Utilizando kernel linear
```

```
modelo_svm_linear = tune.svm(is_attributed ~ app +
                             channel +
                             hour +
                             device +
                             os +
                             nip_h_dev +
                             nip_day_h,
                             data=train,
                             kernel="linear")
```

```
# Salvando a melhor versão do modelo
```

```
best.linear = modelo_svm_linear$best.model
```

```
## Analisando a performance do modelo
```

```
best.test=predict(best.linear,newdata=test,type="class")
confusionMatrix(best.test,test$is_attributed)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  0   1
```

```
##           0 341 157
```

```
##           1  57 195
```

```
##
```

```
##           Accuracy : 0.7147
```

```
##           95% CI : (0.6809, 0.7468)
```

```
##           No Information Rate : 0.5307
```

```
##           P-Value [Acc > NIR] : < 2.2e-16
```

```
##
```

```
##           Kappa : 0.4176
```

```
##
```

```
## Mcnemar's Test P-Value : 1.31e-11
```

```
##
```

```
##           Sensitivity : 0.8568
```

```
##           Specificity : 0.5540
```

```
##          Pos Pred Value : 0.6847
##          Neg Pred Value : 0.7738
##          Prevalence : 0.5307
##          Detection Rate : 0.4547
##          Detection Prevalence : 0.6640
##          Balanced Accuracy : 0.7054
##
##          'Positive' Class : 0
##
```

```
# Utilizando kernel radial
```

```
modelo_svm_radial = tune.svm(is_attributed ~ app +
                             channel +
                             hour +
                             device +
                             os +
                             nip_h_dev +
                             nip_day_h,
                             data=train,
                             kernel="radial")
```

```
## Analisando a performance do modelo
```

```
best.radial=modelo_svm_radial$best.model
best.test=predict(best.radial,newdata=test,type="class")
confusionMatrix(best.test,test$is_attributed)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##          Reference
```

```
## Prediction  0    1
```

```
##           0 347 141
```

```
##           1  51 211
```

```
##
```

```
##          Accuracy : 0.744
```

```
##          95% CI : (0.7112, 0.7749)
```

```
##    No Information Rate : 0.5307
```

```
##    P-Value [Acc > NIR] : < 2.2e-16
```

```
##
```

```
##          Kappa : 0.4784
```

```
##
```

```
##    McNemar's Test P-Value : 1.336e-10
```

```
##
```

```
##          Sensitivity : 0.8719
```

```
##          Specificity : 0.5994
```

```
##          Pos Pred Value : 0.7111
```

```
##          Neg Pred Value : 0.8053
```

```
##          Prevalence : 0.5307
```

```
##          Detection Rate : 0.4627
```

```
##          Detection Prevalence : 0.6507
```

```
##          Balanced Accuracy : 0.7356
```

```
##
```

```
##          'Positive' Class : 0
```

```
##
```

```
# Chamando a Função para executar o registrar todos os resultados do modelo
mult_rf <- nb_multiple_runs(train, 20)
```

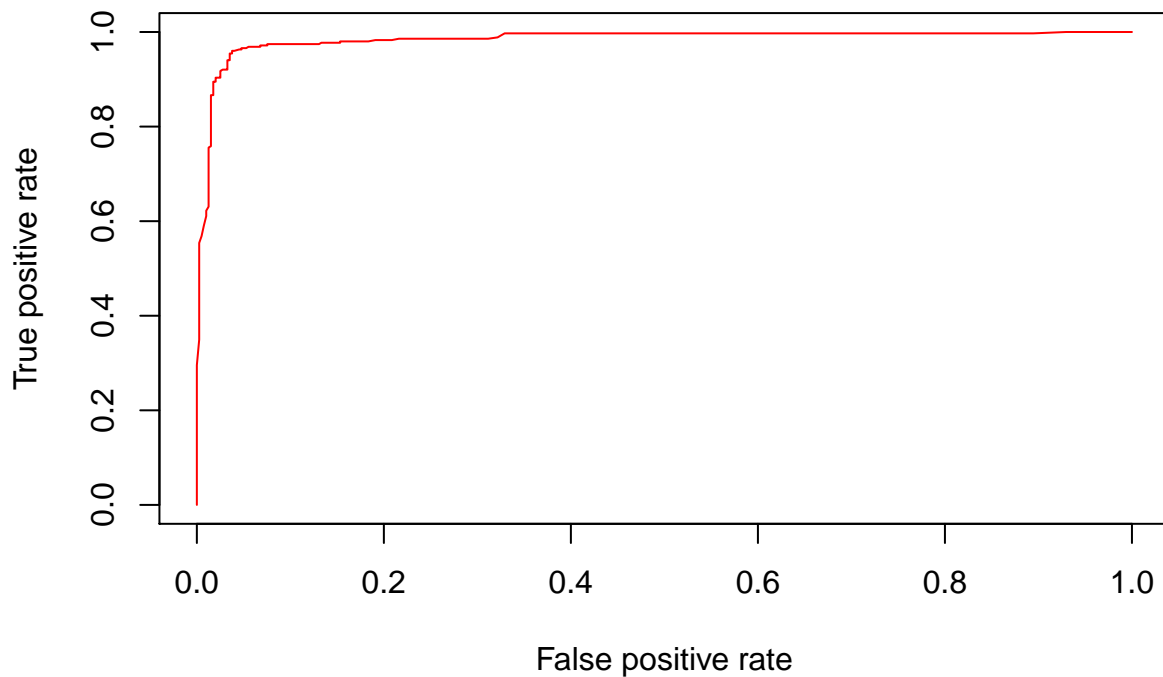
```
# Média de acurácia do modelo
summary(mult_rf)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.9440 0.9530 0.9553 0.9562 0.9590 0.9653
```

Etapa 8 - Curva ROC

```
# Criando curvas ROC
# Gerando as classes de dados
class1 <- predict(modelo_rf2, newdata = test, type = 'prob')
class2 <- test$is_attributed
```

```
# Gerando a curva ROC
pred <- prediction(class1[,2], class2)
perf <- performance(pred, "tpr", "fpr")
plot(perf, col = rainbow(10))
```



Fim! Obrigado.