Predictions whether a user will download an app after clicking a mobile app advertisement

Diogo F. dos Santos

August 9th, 2020

% !TEX encoding = UTF-8 Unicode

PART ONE 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.

Problem: Predict the is_attributed features Data set site: https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data

The solution to this problem was divided into four parts. The first part is in this script. It deals with the data munging and the testing of many machine learning models using the train_sample.csv file and testing with 1E+07 rows of the train.csv. The data of this file was used as the test dataset because the dataset provided did not have the target variable.

The second part of the solution got the main tidying lines of part one to tidy the full training dataset, nominated train.csv. In the third part, the tidying training dataset was taken with the best model acquired in part one to train the model, but the number of the trees of the random forest model was reduced due to my notebook capacity. In the fourth part, the trained model was applied to the provided test dataset, test.csv. Afterward, the predicted results were matched with the click_id to produce the submission file.

```r
# Removes all existing objects and packages from the current workspace
# rm(list = ls())
# Working directory
setwd("~/Documents/learning_Data_Science/R_learnings/Project_1_in_R")
# getwd()

# Packages
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
```

```r
library(lubridate)
```

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

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

```r
library(ggplot2)
library(ggthemes)
library(mltools)
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
```

```r
library(caret)
```

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

```r
library(ROCR)
library(knitr)
library(rmarkdown)
```

```r
# Read the data sets
train_set <- read.csv(file = 'train_sample.csv', header = T)
#test_set <- fread(file = 'test.csv', header = T)

# The train dataset named train.csv can be found on the web site
# https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data
test_set <- fread(file = 'train.csv', header = T, nrows = 1e7)
```

```r
########## Exploratory data analysis ##########

# Missing values
any(is.na(train_set))
```

```
## [1] FALSE
```

```r
any(is.na(test_set))
```

```
## [1] FALSE
```

```r
# Overview
dim(train_set)
```

```
## [1] 100000      8
```

```r
head(train_set)
```

```
##       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
```

```r
str(train_set)
```

```
## '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     : chr  "2017-11-07 09:30:38" "2017-11-07 13:40:27" "2017-11-07 18:05:24" "2017-11-0
##  $ attributed_time: chr  "" "" "" "" ...
##  $ is_attributed  : int  0 0 0 0 0 0 0 0 0 0 ...
```

```r
dim(test_set)
```

```
## [1] 10000000        8
```

```r
head(test_set)
```

```
##        ip app device os channel          click_time attributed_time
## 1:  83230   3      1 13     379 2017-11-06 14:32:21
## 2:  17357   3      1 19     379 2017-11-06 14:33:34
## 3:  35810   3      1 13     379 2017-11-06 14:34:12
## 4:  45745  14      1 13     478 2017-11-06 14:34:52
## 5: 161007   3      1 13     379 2017-11-06 14:35:08
## 6:  18787   3      1 16     379 2017-11-06 14:36:26
##    is_attributed
## 1:             0
## 2:             0
## 3:             0
## 4:             0
## 5:             0
## 6:             0
```

```r
str(test_set)
```

```
## Classes 'data.table' and 'data.frame':   10000000 obs. of  8 variables:
##  $ ip             : int  83230 17357 35810 45745 161007 18787 103022 114221 165970 74544 ...
##  $ app            : int  3 3 3 14 3 3 3 3 3 64 ...
##  $ device         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ os             : int  13 19 13 13 13 16 23 19 13 22 ...
##  $ channel        : int  379 379 379 478 379 379 379 379 379 459 ...
##  $ click_time     : chr  "2017-11-06 14:32:21" "2017-11-06 14:33:34" "2017-11-06 14:34:12" "2017-11-0
```

```
##  $ attributed_time: chr  "" "" "" "" ...
##  $ is_attributed  : int  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
# The target variable is categorical
train_set$is_attributed <- as.factor(train_set$is_attributed)
test_set$is_attributed <- as.factor(test_set$is_attributed)
table(train_set$is_attributed)
```

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

```r
# table(test_set$is_attributed)
                                        # It has other categorical ip, variables,
                                        # like the app, device, os, and channel,
                                        # but it seems to be not practical to
                                        # convert these variables at this time.

# Train dataset summary
summary(train_set)
```

```
##        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      0:99773
##  1st Qu.:145.0   Class :character   Class :character   1:  227
##  Median :258.0   Mode  :character   Mode  :character
##  Mean   :268.8
##  3rd Qu.:379.0
##  Max.   :498.0
```

```r
summary(test_set)
```

```
##        ip              app              device              os
##  Min.   :     9   Min.   :  0.00   Min.   :   0.00   Min.   :  0.0
##  1st Qu.: 42164   1st Qu.:  3.00   1st Qu.:   1.00   1st Qu.: 13.0
##  Median : 81973   Median : 12.00   Median :   1.00   Median : 18.0
##  Mean   : 87332   Mean   : 12.86   Mean   :  33.04   Mean   : 24.6
##  3rd Qu.:121187   3rd Qu.: 15.00   3rd Qu.:   1.00   3rd Qu.: 19.0
##  Max.   :212774   Max.   :675.00   Max.   :3545.00   Max.   :745.0
##     channel        click_time         attributed_time    is_attributed
##  Min.   :  0.0   Length:10000000    Length:10000000    0:9981283
##  1st Qu.:134.0   Class :character   Class :character   1:  18717
##  Median :237.0   Mode  :character   Mode  :character
##  Mean   :252.7
##  3rd Qu.:377.0
##  Max.   :498.0
```

```r
# Unique values of the ip feature
length(unique(train_set$ip))        # 34857 values in 100000 of the total.
```

```
## [1] 34857
```

```r
length(unique(test_set$ip))          # 93936 values in 18790469 of the total.
```

```
## [1] 68740
```

```r
head(rev(sort(table(train_set$ip))))
```

```
##
##   5348   5314  73487  73516  53454 114276
##    669    616    439    399    280    219
```

```r
head(rev(sort(table(test_set$ip)))) # Waw!!! It has many ip repetitions.
```

```
##
##  73516  73487   5314   5348  53454 105560
##  51711  51215  35073  35004  25381  23289
```

```r
                                    # I thought it had had much less than that.
                                    # Some ips have so many repetitions that
                                    # I think I will have to make classes to
                                    # compute this dependency and analyze if
                                    # the target variable has a strong
                                    # dependency on the ip variable.
                                    # Maybe classes depending on the number
                                    # of repetitions.
```

```r
# Duplicated ips
dupl_ips_train <- train_set[duplicated(train_set$ip), 1]
length(dupl_ips_train)
```

```
## [1] 65143
```

```r
length(unique(dupl_ips_train))
```

```
## [1] 17434
```

```r
round(prop.table(table(train_set$is_attributed[train_set$ip %in%
                                unique(dupl_ips_train)])) * 100, 2)
```

```
##
##     0     1
## 99.91  0.09
```

```r
dupl_ips_test <- train_set[duplicated(test_set$ip), 1]
length(dupl_ips_test)
```

```
## [1] 9931260
```

```r
length(unique(dupl_ips_test))
```

```
## [1] 32051
```

```r
round(prop.table(table(train_set$is_attributed[train_set$ip %in%
                                unique(dupl_ips_test)])) * 100, 2)
```

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

```r
# Repeated ips in order
n_dupl_ips_train <- train_set %>%
  count(ip, wt = n() ) %>%
  arrange(desc(n))

head(n_dupl_ips_train)
```

```
##        ip   n
## 1    5348 669
## 2    5314 616
## 3   73487 439
## 4   73516 399
## 5   53454 280
## 6  114276 219
```

```r
n_dupl_ips_test <- test_set %>%
  count(ip, wt = n() ) %>%
  arrange(desc(n))

head(n_dupl_ips_test)
```

```
##         ip     n
## 1:   73516 51711
## 2:   73487 51215
## 3:    5314 35073
## 4:    5348 35004
## 5:   53454 25381
## 6:  105560 23289
```

```r
# Verifyind the total of lines
sum(n_dupl_ips_train$n)
```

```
## [1] 100000
```

```r
sum(n_dupl_ips_test$n)
```

```
## [1] 10000000
```

```r
# Number of duplicate ips column
train_set <- left_join(train_set, n_dupl_ips_train, by = 'ip')
head(train_set)
```

```
##        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   n
## 1             0   8
## 2             0 149
## 3             0   2
## 4             0   3
## 5             0   4
## 6             0   2
```

```r
test_set <- left_join(test_set, n_dupl_ips_test, by = 'ip')
head(test_set)
```

```
##        ip app device os channel          click_time attributed_time
## 1:  83230   3      1 13     379 2017-11-06 14:32:21
## 2:  17357   3      1 19     379 2017-11-06 14:33:34
## 3:  35810   3      1 13     379 2017-11-06 14:34:12
## 4:  45745  14      1 13     478 2017-11-06 14:34:52
## 5: 161007   3      1 13     379 2017-11-06 14:35:08
## 6:  18787   3      1 16     379 2017-11-06 14:36:26
##    is_attributed    n
## 1:             0 1327
## 2:             0 1057
## 3:             0  449
## 4:             0 9395
## 5:             0  184
## 6:             0  205
```

```r
# Rename the n columns
names(train_set)[9] <- 'repetitions'
labels(train_set)[[2]]
```

```
## [1] "ip"              "app"             "device"          "os"
## [5] "channel"         "click_time"      "attributed_time" "is_attributed"
## [9] "repetitions"
```

```r
names(test_set)[9] <- 'repetitions'
# names(test_set)[8] <- 'repetitions'
labels(test_set)[[2]]
```

```
## [1] "ip"              "app"             "device"          "os"
## [5] "channel"         "click_time"      "attributed_time" "is_attributed"
## [9] "repetitions"
```

```r
# The number of ips repeated depending on the number of repetitions
c = 1
values <- unique(n_dupl_ips_train$n)
df <- data.frame(repetitions = rep(NA, length(values)))

for (i in values) {
  df$repetitions[c] <- i
  tab <- table(train_set$is_attributed[train_set$repetitions == i])
  df$no[c] <- tab[1]
  df$no_prop[c] <- round(tab[1] * 100 / sum(tab), 2)
  df$yes[c] <- tab[2]
  df$yes_prop[c] <- round(tab[2] * 100 / sum(tab), 2)
  c = c + 1
}

# Verifying the number rows of  train data set is correct
sum(df$no, df$yes)
```

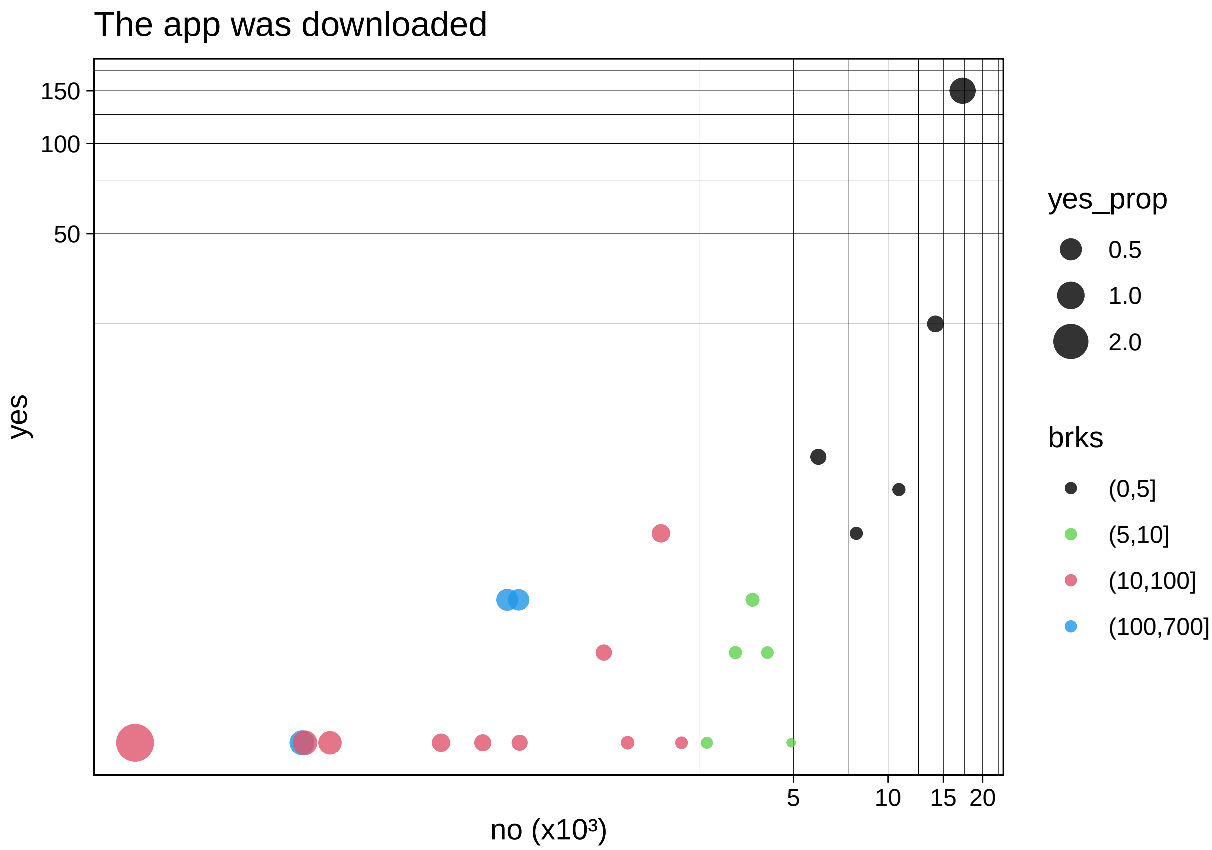
```
## [1] 100000
```

```r
# Filter and sorting df in relation to the proportion of yes to the app
df_prop <- df %>%
  filter(yes_prop > 0) %>%
```

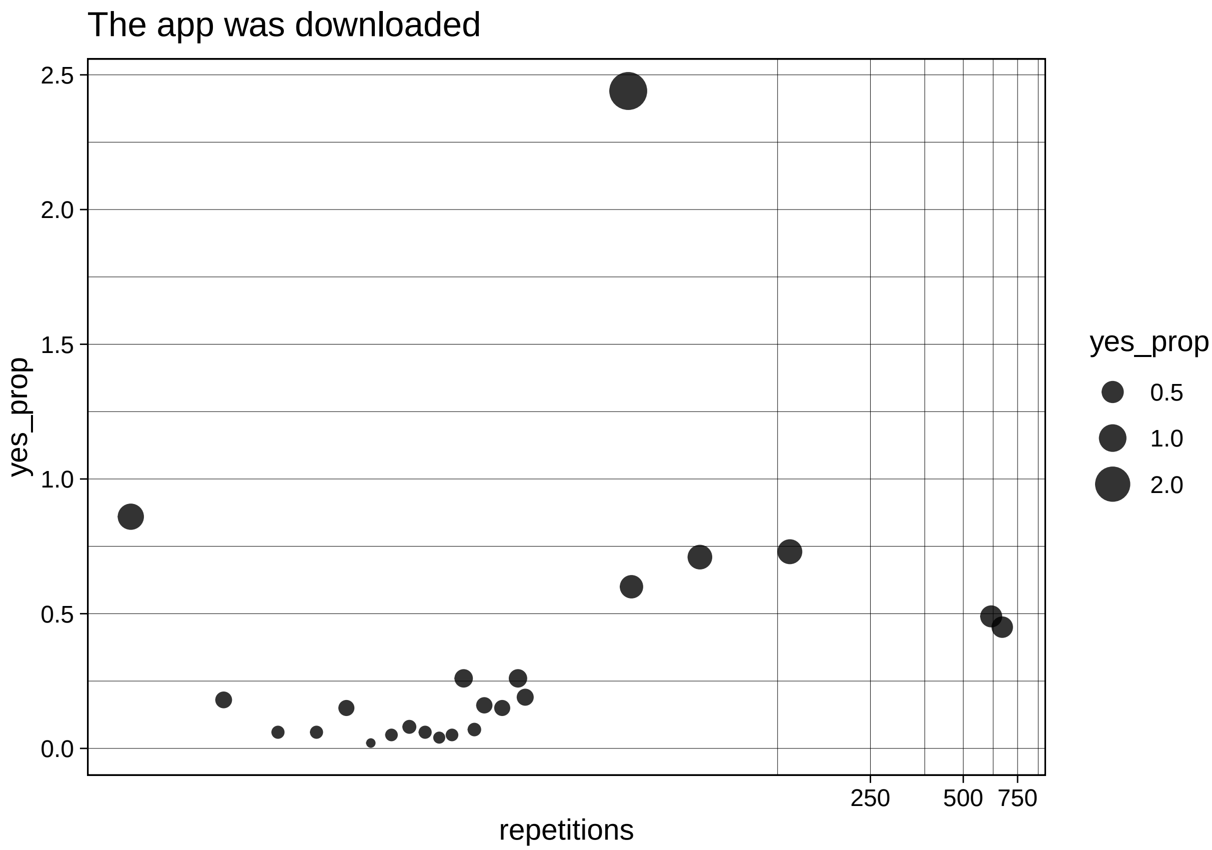
```r
  arrange(desc(yes_prop))

df_prop
```

```
##    repetitions    no no_prop yes yes_prop
## 1           41    40   97.56   1     2.44
## 2            1 17273   99.14 150     0.86
## 3          137   136   99.27   1     0.73
## 4           70   139   99.29   1     0.71
## 5           42   167   99.40   1     0.60
## 6          616   613   99.51   3     0.49
## 7          669   666   99.55   3     0.45
## 8           18   377   99.74   1     0.26
## 9           12  1891   99.74   5     0.26
## 10          19   512   99.81   1     0.19
## 11           2 14153   99.82  25     0.18
## 12          14  1244   99.84   2     0.16
## 13          16   671   99.85   1     0.15
## 14           5  5996   99.85   9     0.15
## 15           8  3701   99.92   3     0.08
## 16          13  1481   99.93   1     0.07
## 17           9  3265   99.94   2     0.06
## 18           4  7923   99.94   5     0.06
## 19           3 10826   99.94   7     0.06
## 20          11  2199   99.95   1     0.05
## 21           7  4128   99.95   2     0.05
## 22          10  2649   99.96   1     0.04
## 23           6  4913   99.98   1     0.02
```

```r
# Scatter plot of the yes/no downloading app and the number of ips repetitions
brks <- cut(df_prop$repetitions, breaks = c(0, 5, 10, 100, 700))
ggplot(data = df_prop) +
  geom_point(aes(no/1000, yes, color = brks,
                 size = yes_prop), alpha = 0.8) +
  xlab('no (x10³)') +
  scale_color_manual(values = c(1,3,2,4)) +
  scale_size(breaks = c(0.5, 1, 2)) +
  coord_trans(x = 'log', y = 'log') +
  ggtitle('The app was downloaded') +
  theme_linedraw()
```




```r
# Scatter plot of the yes/no downloading app and the number of ips repetitions
ggplot(data = df_prop) +
  geom_point(aes(repetitions, yes_prop, size = yes_prop), alpha = 0.8) +
  scale_color_manual(values = c(1,3,2,4)) +
  scale_size(breaks = c(0.5, 1, 2)) +
  coord_trans(x = 'log') +
  ggtitle('The app was downloaded') +
  theme_linedraw()
```




```
                                      # the yes proportions seems to behave like
                                      # a sinusoid

# Iserting anoter columns according to the yes_prop values
gt <- df_prop$repetitions[df_prop$yes_prop > 0.4]

train_set <- train_set %>%
  mutate(yes_prop = ifelse(repetitions %in% gt, 1, 0))
                                      # I forget that did not have the
                                      # is_attributed features in test data set.
                                      # I will make classes for repititions
                                      # feature.

# repetitions classes
train_set$repetitions_fac <- cut(train_set$repetitions,
                                 breaks = c(0,5,nrow(train_set)),
                                 labels = c(1, 2))

test_set$repetitions_fac <- cut(test_set$repetitions,
                                breaks = c(0,5,nrow(test_set)),
                                labels = c(1, 2))

#########  TIME VARIABLE  ##################

# The click_time feature of the train data set
train_set$click_time <- as.Date(train_set$click_time, format = '%Y-%m-%d')
unique(months(train_set$click_time))      # only in november
```

```
## [1] "novembro"
```

```r
unique(year(train_set$click_time))         # only in 2017
```

```
## [1] 2017
```

```r
unique(day(train_set$click_time))          # the days are between 6 and 9
```

```
## [1] 7 9 8 6
```

```r
unique(weekdays(train_set$click_time))     # days 6 = Monday, 7 = Tuesday,
```

```
## [1] "terça"   "quinta"  "quarta"  "segunda"
```

```r
                                           #      8 = Wednesday, and 9 = Thursday

# The click_time feature of the test data set
test_set$click_time <- as.Date(test_set$click_time, format = '%Y-%m-%d')
unique(months(test_set$click_time))        # only in november
```

```
## [1] "novembro"
```

```r
unique(year(test_set$click_time))          # only in 2017
```

```
## [1] 2017
```

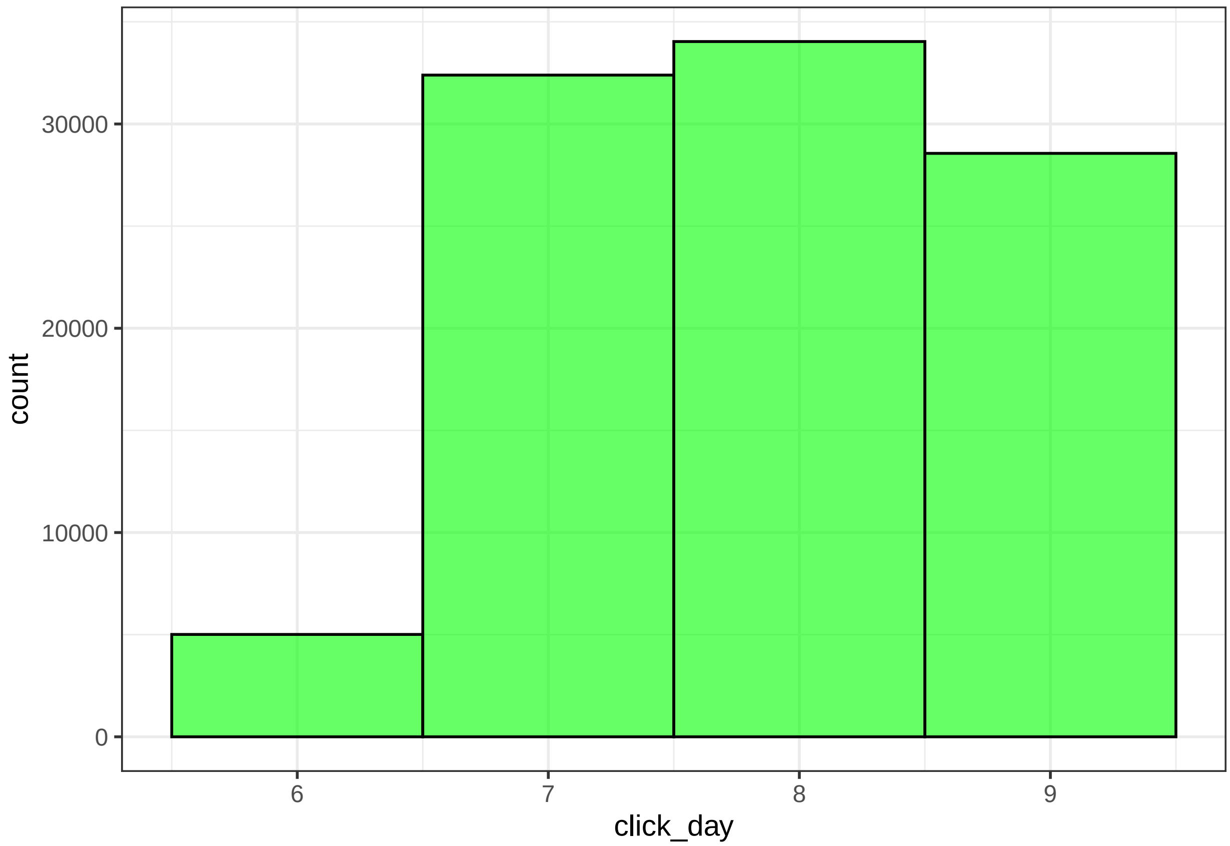
```r
unique(day(test_set$click_time))           # only the day 10th
```

```
## [1] 6 7
```

```r
unique(weekdays(test_set$click_time))      # day  10 = friday
```

```
## [1] "segunda" "terça"
```

```r
# New feature containing the day of the click_time
train_set$click_day <- day(train_set$click_time)
test_set$click_day <- day(test_set$click_time)

# train click_day plot
ggplot(train_set, aes(click_day)) +
  geom_histogram(binwidth = 1, fill = 'green', col = 'black', alpha = 0.6) +
  theme_bw()
```

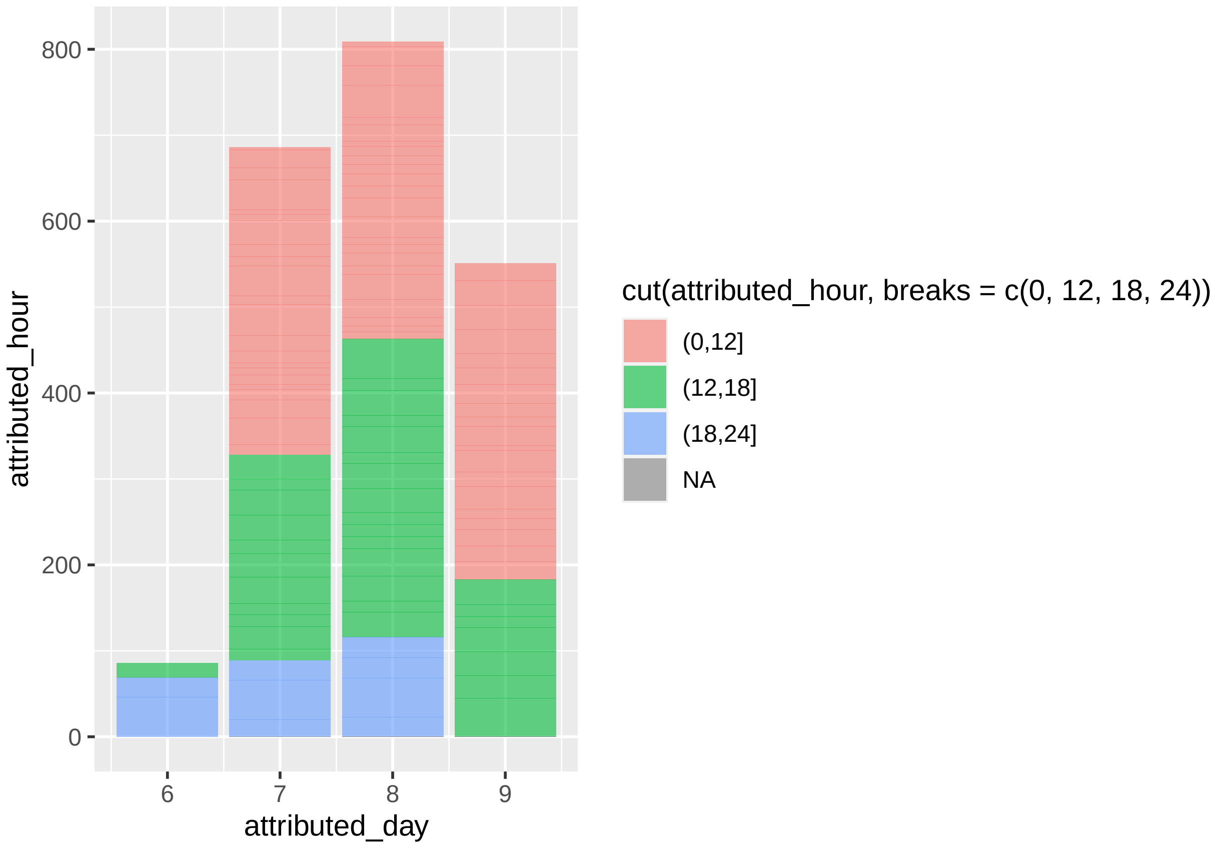



```r
                                            # maybe the day 6 has only the
                                            # partial data

# Number of clicks in function of the day (train)
train_set %>%
  count(click_day)
```

```
##   click_day     n
## 1         6  5011
## 2         7 32393
## 3         8 34035
## 4         9 28561
```

```r
# The train attributed_time feature. It is not in the test data set.
train_set$attributed_time <- ymd_hms(train_set$attributed_time)

# New features containing the day and hour of the train attributed_time
train_set$attributed_day <- day(train_set$attributed_time)
train_set$attributed_hour <- hour(train_set$attributed_time) +
  ifelse(minute(train_set$attributed_time) >= 30, 1, 0)

ggplot(train_set, aes(attributed_day, attributed_hour,
                      fill = cut(attributed_hour, breaks = c(0,12,18,24)))) +
  geom_bar(stat = "identity", alpha = 0.6)
```

```
## Warning: Removed 99773 rows containing missing values (position_stack).
```

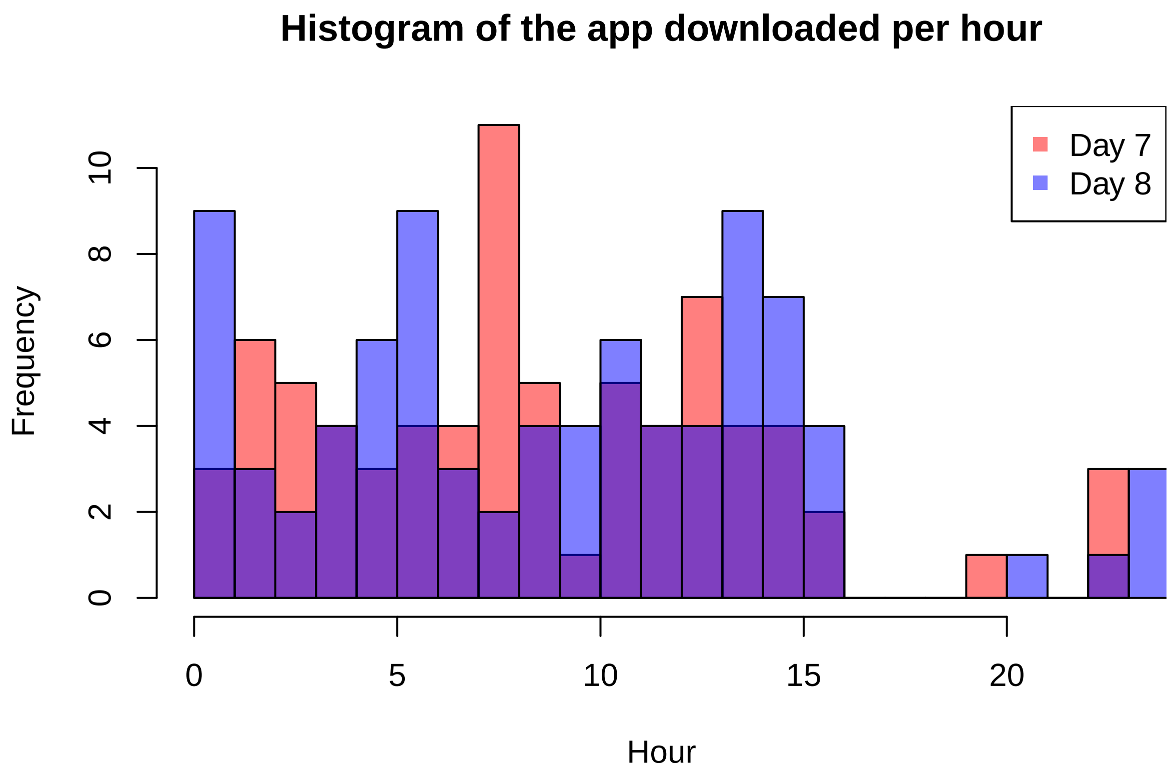



```r
# Number of clicks in function of the day (train)
train_set %>%
  count(attributed_day)
```

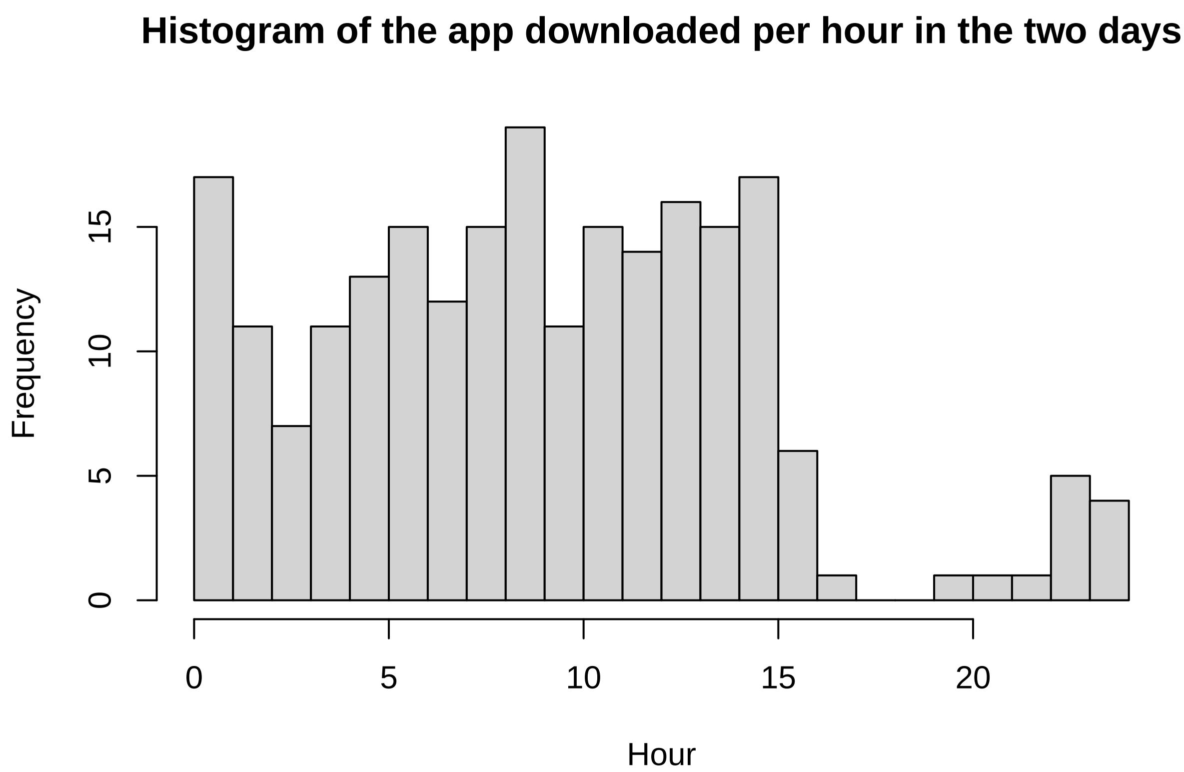
```
##   attributed_day     n
## 1              6     4
## 2              7    76
## 3              8    85
## 4              9    62
## 5             NA 99773
```

```r
                                                           # As shown in the last results, days 6
                                                           # and 9 have fewer observations than
                                                           # the other days. It seems that the
                                                           # observations of the day 6 were in
                                                           # the final of the afternoon and
                                                           # day 9 until the middle of the
                                                           # afternoon. I will eliminate the
                                                           # day 6 and 9 to have two entire days.
                                                           # This data set dos not have much
                                                           # positive targets. I will not
                                                           # delete the day 6 and 9.

# Erase the day 6 and 9 to have two entire days (train)
# train_set <- train_set[train_set$click_day != 6 & train_set$click_day != 9, ]
# dim(train_set)
```

```
# Hour of the day that the app was downloaded
hist(train_set$attributed_hour[train_set$attributed_day == 7],
     col = rgb(1,0,0,0.5), breaks = 24,
     main = 'Histogram of the app downloaded per hour', xlab = 'Hour')
hist(train_set$attributed_hour[train_set$attributed_day == 8],
     col = rgb(0,0,1,0.5), breaks = 24, add = T)
legend(x = "topright", legend = c('Day 7','Day 8'),
       col = c(rgb(1,0,0,0.5), rgb(0,0,1,0.5)), pch = 15)
```




```
hist(train_set$attributed_hour,
     breaks = 24, main = 'Histogram of the app downloaded per hour in the two days',
     xlab = 'Hour')
```




```r
                                                # Strangely, the number of downloads
                                                # has a great decrease after 16 hours


# app feature
sort(unique(train_set$app))
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  42  43  44  45  46  47  48  49  50  52  53  54  55  56  58
##  [55]  59  60  61  62  64  65  66  67  68  70  71  72  74  75  76  78  79  80
##  [73]  81  82  83  84  85  86  87  88  91  92  93  94  95  96  97  99 100 101
##  [91] 103 104 105 107 108 109 110 112 115 116 117 118 119 121 122 123 124 125
## [109] 133 134 137 139 145 146 148 149 150 151 158 160 161 163 165 168 170 171
## [127] 176 181 183 190 192 202 204 208 215 216 232 233 261 266 267 268 271 273
## [145] 293 302 310 315 347 363 372 394 398 407 425 474 486 536 538 548 551
```

```r
sort(unique(test_set$app))
```

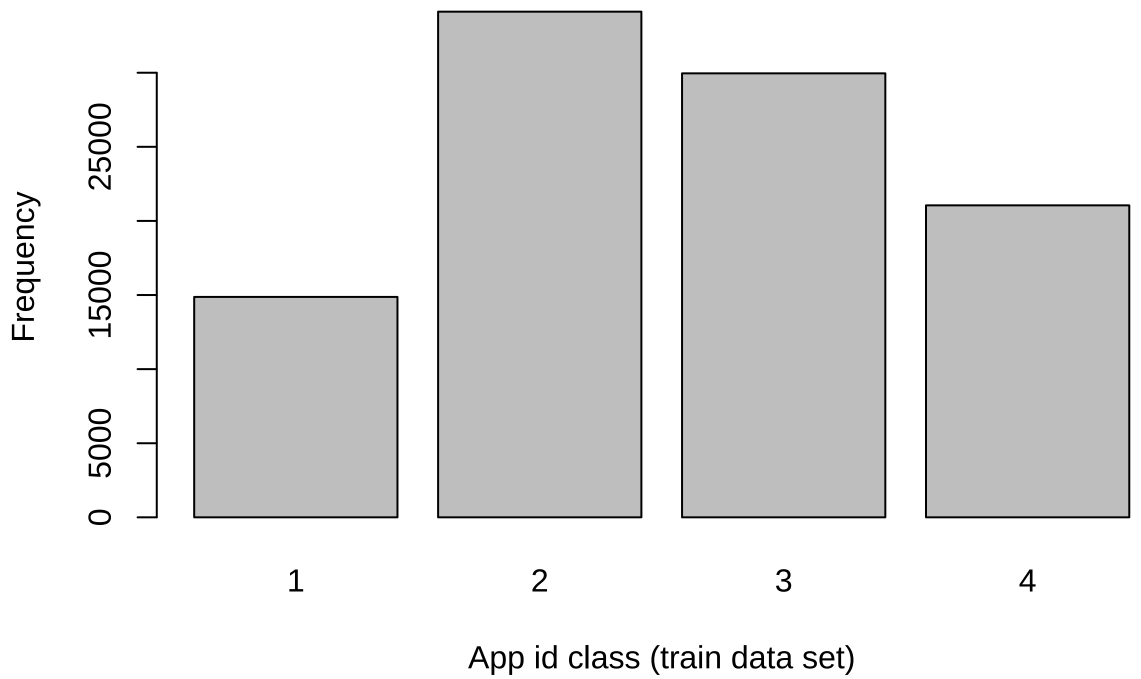
```
##   [1]   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##  [19]  18  19  20  21  22  23  24  25  26  27  28  29  32  33  34  35  36  37
##  [37]  38  39  40  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56
##  [55]  57  58  59  60  61  62  64  65  66  67  68  69  70  71  72  73  74  75
##  [73]  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  93  94
##  [91]  95  96  97  98  99 100 101 102 103 104 105 107 108 109 110 111 112 114
## [109] 115 116 118 119 120 121 122 123 124 125 126 127 128 130 132 133 134 136
## [127] 137 140 141 142 143 145 146 148 149 150 151 152 153 154 155 158 159 160
## [145] 161 162 163 165 166 167 168 169 170 171 172 173 175 176 177 181 182 183
## [163] 184 185 186 188 190 192 193 194 195 196 197 198 199 202 203 205 206 207
## [181] 208 209 210 212 213 215 216 217 218 220 222 223 224 226 229 230 231 232
## [199] 233 236 237 238 239 240 241 242 244 246 247 249 250 251 255 256 257 258
## [217] 259 261 262 263 265 266 267 268 269 272 273 276 277 278 279 280 281 283
```

```
## [235] 284 286 288 289 290 291 292 294 295 299 302 303 304 305 310 312 315 317
## [253] 318 319 320 322 324 325 326 328 329 333 334 336 346 347 349 352 354 355
## [271] 361 362 363 365 366 367 371 372 376 379 381 383 386 394 395 398 399 407
## [289] 419 425 429 433 436 443 446 448 469 474 480 481 484 489 496 502 525 530
## [307] 531 536 537 538 540 541 549 551 553 555 556 557 561 563 564 565 569 576
## [325] 610 612 619 625 629 645 651 675
```

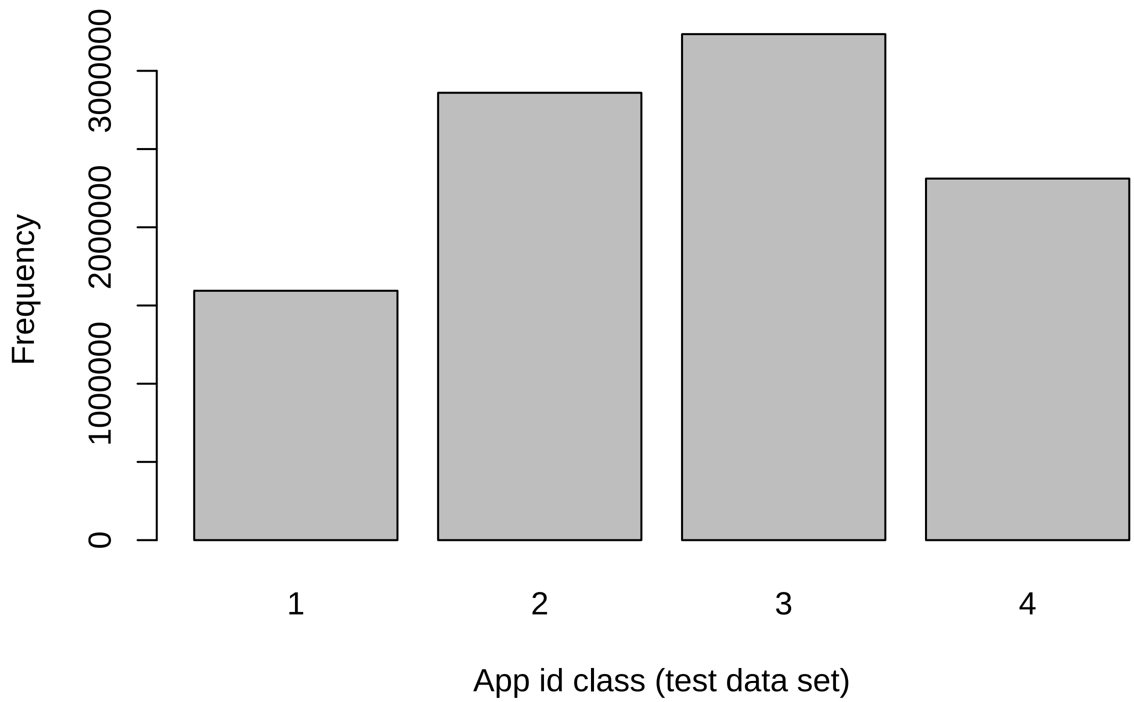
```r
div_app<- bin_data(c(train_set$app, test_set$app), bins = 4, binType = "quantile")
levels(div_app)
```

```
## [1] "[0, 3)"    "[3, 12)"   "[12, 15)"  "[15, 675]"
```

```r
train_set$app_fac <- cut(train_set$app, breaks = c(0, 3, 12, 18, nrow(train_set)),
                    right = F, labels = c(1, 2, 3, 4))

test_set$app_fac <- cut(test_set$app, breaks = c(0, 3, 12, 18, nrow(test_set)),
                    right = F, labels = c(1, 2, 3, 4))


plot(train_set$app_fac, xlab = 'App id class (train data set)', ylab = 'Frequency')
```




```r
plot(test_set$app_fac, xlab = 'App id class (test data set)', ylab = 'Frequency')
```




```r
# device feature
sort(unique(train_set$device))
```

```
##   [1]    0    1    2    4    5    6    7    9   11   16   17   18   20   21   25
##  [16]   30   33   36   37   40   49   50   53   56   58   59   60   67   74   76
##  [31]   78   79   97  100  102  103  106  109  114  116  124  129  154  160  163
##  [46]  167  180  182  188  196  202  203  210  211  220  241  268  291  321  329
##  [61]  347  351  362  374  385  386  414  420  486  516  549  552  558  579  581
##  [76]  596  607  657  828  883  928  957 1042 1080 1162 1318 1422 1482 1728 1839
##  [91] 2120 2429 2980 3032 3282 3331 3543 3545 3866 3867
```

```r
sort(unique(test_set$device))
```

```
##   [1]    0    1    2    4    6    7    8    9   10   11   13   14   15   16   17
##  [16]   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
##  [31]   33   34   35   36   37   38   39   40   42   43   44   45   46   48   49
##  [46]   50   51   52   53   54   56   57   58   59   60   61   62   63   64   65
##  [61]   66   67   68   70   71   72   73   74   75   76   77   78   79   80   81
##  [76]   82   83   84   86   88   89   90   92   93   94   95   96   97   99  100
##  [91]  101  102  103  104  105  106  107  108  109  110  112  113  114  115  116
## [106]  118  119  122  123  124  126  127  128  129  130  131  132  133  134  136
## [121]  137  138  140  141  142  144  145  146  147  148  149  150  151  152  153
## [136]  154  155  156  157  159  160  161  162  163  164  165  166  167  168  169
## [151]  170  171  172  173  174  175  177  179  180  181  183  184  185  186  187
## [166]  188  189  190  191  192  194  195  196  197  199  200  201  202  203  204
## [181]  205  206  207  208  209  210  211  212  213  214  215  217  218  219  220
## [196]  221  224  226  227  228  229  230  231  233  234  235  236  239  240  241
## [211]  242  244  247  248  249  250  251  253  254  256  259  260  261  262  263
## [226]  264  265  266  268  270  271  272  273  274  275  276  277  278  280  284
## [241]  286  287  288  289  290  291  292  294  296  298  299  301  304  305  308
## [256]  309  310  311  312  313  314  315  316  317  319  320  321  322  323  324
## [271]  326  327  328  329  331  333  334  336  337  338  339  340  342  343  344
## [286]  345  346  347  348  349  350  351  353  354  356  357  360  361  362  365
```

```
## [301]  367  369  370  372  374  376  377  379  381  383  384  385  386  387  389
## [316]  391  395  396  397  398  399  400  401  403  405  407  408  409  410  413
## [331]  414  415  416  417  418  419  420  421  422  423  424  426  431  432  433
## [346]  434  435  437  439  441  442  447  450  452  453  454  458  459  460  461
## [361]  462  463  464  465  466  467  468  469  470  472  477  479  481  482  483
## [376]  485  486  490  494  495  496  497  498  500  501  503  504  505  507  511
## [391]  513  514  520  523  526  527  528  529  530  531  532  533  536  537  538
## [406]  540  542  544  545  548  554  555  556  557  558  564  565  566  567  569
## [421]  571  572  573  576  577  578  579  580  581  584  585  586  587  588  591
## [436]  592  596  597  602  605  611  613  616  617  620  621  624  627  628  629
## [451]  635  636  637  638  640  641  643  648  650  651  653  655  657  658  667
## [466]  669  670  671  673  674  685  686  693  697  698  700  701  703  704  706
## [481]  707  708  709  710  711  712  713  715  718  719  720  721  723  724  727
## [496]  728  729  730  734  735  736  737  738  739  743  745  746  747  748  751
## [511]  754  756  759  762  764  765  766  767  768  771  776  783  788  790  791
## [526]  792  795  800  805  806  808  812  818  819  828  830  831  832  839  840
## [541]  844  847  848  850  854  857  858  861  864  869  870  872  875  879  885
## [556]  886  887  892  893  901  903  904  911  913  915  916  919  922  923  924
## [571]  928  929  932  937  941  944  945  948  951  952  954  955  965  968  969
## [586]  970  972  973  976  977  980  983  984 1002 1003 1009 1012 1014 1016 1027
## [601] 1031 1034 1036 1039 1042 1044 1045 1046 1047 1048 1053 1055 1058 1059 1061
## [616] 1066 1067 1076 1077 1079 1080 1086 1088 1089 1093 1100 1101 1111 1115 1117
## [631] 1118 1121 1123 1126 1128 1131 1134 1140 1141 1142 1143 1144 1145 1147 1148
## [646] 1154 1156 1157 1160 1162 1165 1167 1168 1170 1172 1174 1175 1176 1178 1183
## [661] 1184 1193 1195 1199 1202 1208 1209 1214 1215 1218 1219 1226 1227 1228 1230
## [676] 1232 1237 1249 1253 1255 1258 1259 1261 1265 1272 1273 1282 1283 1291 1296
## [691] 1297 1305 1307 1312 1316 1320 1321 1343 1345 1350 1351 1362 1386 1387 1392
## [706] 1396 1397 1398 1401 1404 1408 1410 1419 1422 1423 1428 1429 1436 1449 1451
## [721] 1464 1465 1477 1493 1498 1501 1508 1510 1511 1524 1531 1532 1534 1537 1540
## [736] 1541 1546 1563 1565 1578 1591 1593 1600 1616 1618 1619 1626 1627 1631 1638
## [751] 1653 1672 1680 1690 1694 1697 1711 1719 1726 1728 1730 1734 1735 1745 1772
## [766] 1781 1789 1793 1795 1799 1803 1818 1819 1826 1829 1835 1837 1839 1846 1854
## [781] 1861 1864 1872 1873 1897 1909 1922 1929 1950 1962 1967 1971 1974 2018 2024
## [796] 2030 2036 2045 2050 2070 2075 2081 2093 2104 2115 2147 2174 2195 2204 2206
## [811] 2217 2218 2242 2248 2255 2289 2300 2333 2346 2351 2371 2375 2385 2388 2403
## [826] 2411 2421 2424 2426 2427 2430 2464 2503 2533 2540 2546 2564 2576 2583 2649
## [841] 2653 2663 2664 2703 2706 2731 2743 2766 2774 2787 2795 2813 2815 2818 2849
## [856] 2854 2886 2888 2898 2917 2930 2953 2985 3014 3021 3023 3032 3033 3036 3039
## [871] 3042 3045 3050 3055 3068 3081 3082 3085 3086 3093 3097 3098 3100 3102 3103
## [886] 3121 3124 3143 3146 3147 3158 3160 3161 3164 3168 3173 3191 3199 3201 3232
## [901] 3239 3248 3251 3273 3283 3296 3306 3311 3316 3320 3323 3331 3338 3341 3349
## [916] 3358 3365 3373 3378 3379 3391 3430 3434 3460 3463 3480 3481 3482 3489 3493
## [931] 3503 3507 3512 3518 3522 3524 3525 3527 3537 3545
```

```r
summary(train_set$device)
```

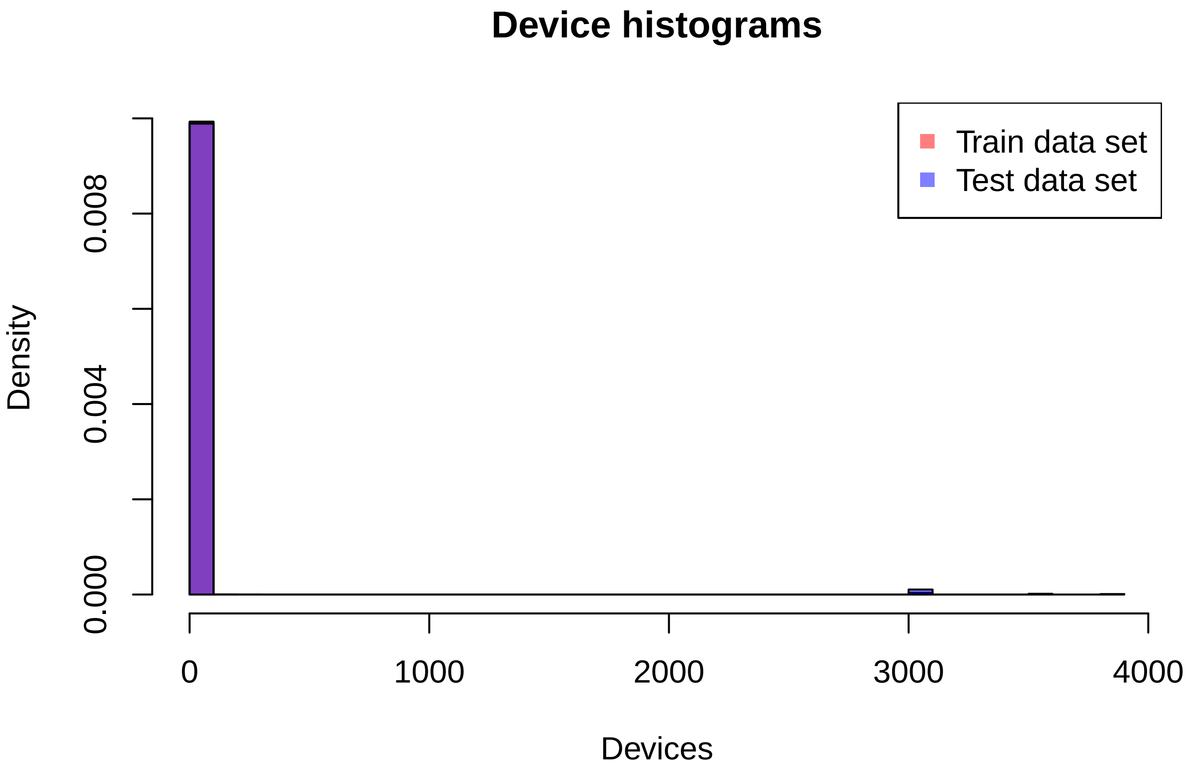
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.00    1.00   21.77    1.00 3867.00
```

```r
summary(test_set$device)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.00    1.00   33.04    1.00 3545.00
```

```r
hist(train_set$device, freq = F, breaks = 40, col = rgb(1,0,0,0.5),
     main = 'Device histograms', xlab = 'Devices')
```

```r
hist(test_set$device, freq = F, breaks = 40, col = rgb(0,0,1,0.5), add = T)
legend(x = "topright", legend = c('Train data set','Test data set'),
       col = c(rgb(1,0,0,0.5), rgb(0,0,1,0.5)), pch = 15)
```




```r
a <- train_set %>%
  count(device, sort = T)
head(a)
```

```
##   device     n
## 1      1 94338
## 2      2  4345
## 3      0   541
## 4   3032   371
## 5   3543   151
## 6   3866    93
```

```r
b <- test_set %>%
  count(device, sort = T)
head(b)
```

```
##     device       n
## 1:       1 9381146
## 2:       2  456617
## 3:    3032  104393
## 4:       0   46476
## 5:      59    1618
## 6:      40     462
```

```r
# Type 1 device proportion
( a[1,2]/sum(a) )                           # 58% of all devices are of type 1
```

```
## [1] 0.5842556
```

```r
( b[1,2]/sum(b) )                           # 81% of all devices are of type 1
```

```
##              n
## 1: 0.8574944
```

```r
# Making two classes of devices: one for type 1 and the other for the others
class_device <- function(x) {ifelse(x == 1, 1, 2)}

train_set$device_fac <- as.factor(class_device(train_set$device))
levels(train_set$device_fac)
```

```
## [1] "1" "2"
```

```r
test_set$device_fac <- as.factor(class_device(test_set$device))
levels(test_set$device_fac)
```

```
## [1] "1" "2"
```

```r
# OS feature
sort(unique(train_set$os))
```

```
##   [1]   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##  [19]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  52  53  55  56
##  [55]  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  73  74  76
##  [73]  77  78  79  80  81  83  84  85  87  88  90  92  96  97  98  99 100 102
##  [91] 106 107 108 109 110 111 112 113 114 116 117 118 127 129 132 133 135 137
## [109] 138 142 151 152 153 155 168 172 174 178 184 185 192 193 196 198 199 207
## [127] 607 748 836 866
```

```r
sort(unique(test_set$os))
```

```
##   [1]   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##  [19]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  52  53  54  55
##  [55]  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73
##  [73]  74  75  76  77  78  79  80  81  83  84  85  86  87  88  89  90  91  92
##  [91]  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110
## [109] 111 112 113 114 115 117 118 119 120 123 124 125 126 127 128 129 130 132
## [127] 133 134 135 136 137 138 140 141 142 143 145 146 147 148 149 150 151 152
## [145] 153 155 156 158 159 160 161 162 164 168 169 171 172 173 174 175 177 178
## [163] 183 184 185 188 190 192 193 196 197 198 207 208 209 213 214 215 216 217
## [181] 219 223 226 228 229 231 234 236 237 241 243 244 245 248 250 251 252 254
## [199] 255 256 260 261 262 265 268 272 274 277 280 284 286 294 297 300 302 305
## [217] 306 325 326 329 336 338 342 346 355 380 404 407 408 411 414 421 438 465
## [235] 505 508 512 514 531 541 552 559 566 573 584 602 603 607 610 612 616 617
## [253] 619 620 622 630 636 640 645 647 649 651 653 656 657 669 672 675 681 684
## [271] 686 687 688 690 692 700 701 702 704 705 707 712 715 716 726 736 737 739
## [289] 742 743 744 745
```

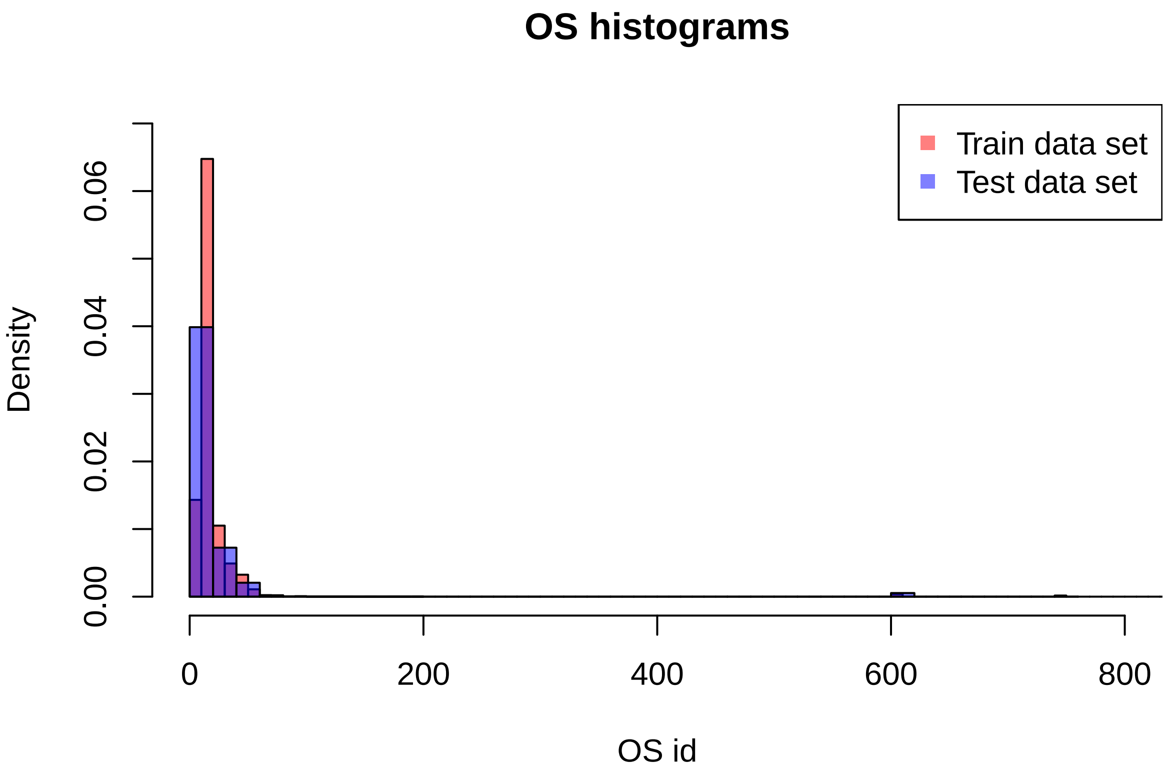
```r
summary(train_set$os)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   13.00   18.00   22.82   19.00  866.00
```

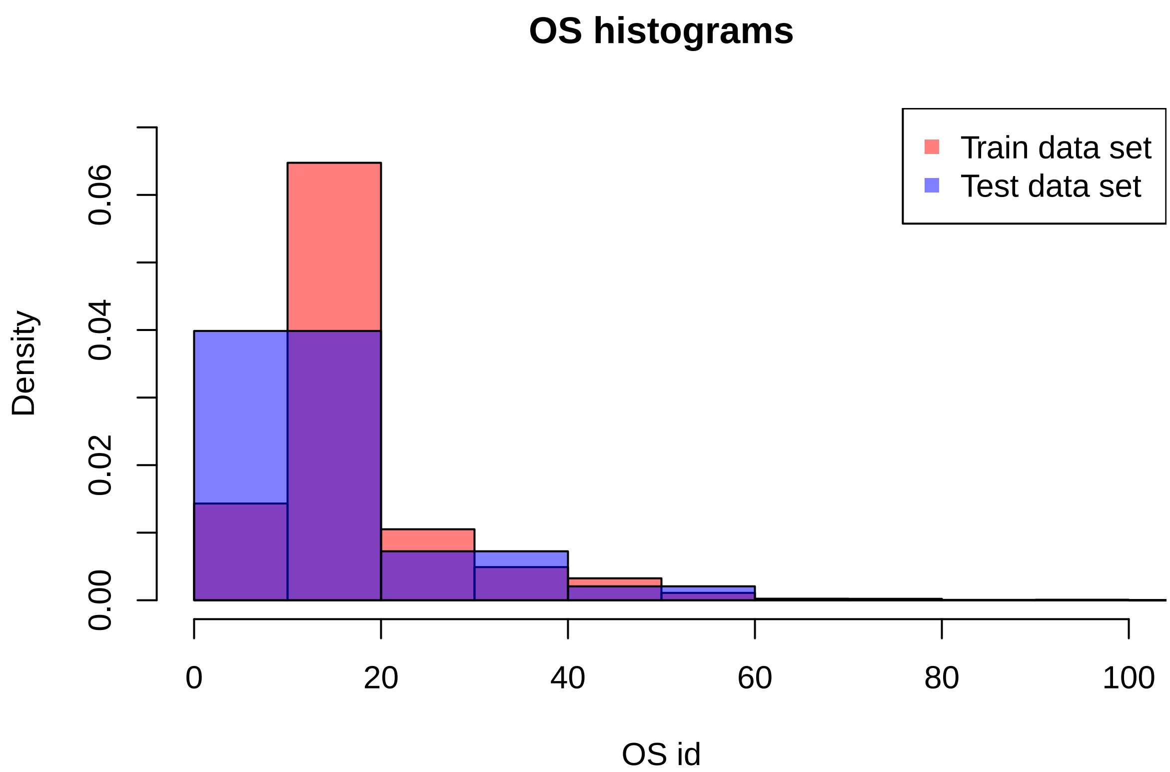
```r
summary(test_set$os)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    13.0    18.0    24.6    19.0   745.0
```

```r
# histograms
hist(train_set$os, freq = F, xlim = c(0,800), ylim = c(0, 0.07), breaks = 100,
     col = rgb(1,0,0,0.5), main = 'OS histograms', xlab = 'OS id')
hist(test_set$os, freq = F, xlim = c(0,800), breaks = 50,
     col = rgb(0,0,1,0.5), add = T)
legend(x = "topright", legend = c('Train data set','Test data set'),
       col = c(rgb(1,0,0,0.5), rgb(0,0,1,0.5)), pch = 15)
```




```r
# smaller domain
hist(train_set$os, freq = F, xlim = c(0, 100), ylim = c(0, 0.07), breaks = 100,
     col = rgb(1,0,0,0.5), main = 'OS histograms', xlab = 'OS id')
hist(test_set$os, freq = F, xlim = c(0,100), breaks = 50,
     col = rgb(0,0,1,0.5), add = T)
legend(x = "topright", legend = c('Train data set','Test data set'),
       col = c(rgb(1,0,0,0.5), rgb(0,0,1,0.5)), pch = 15)
```




```r
# Countings
a <- train_set %>%
  count(os, sort = T)
head(a)
```

```
##   os     n
## 1 19 23870
## 2 13 21223
## 3 17  5232
## 4 18  4830
## 5 22  4039
## 6 10  2816
```

```r
b <- test_set %>%
  count(os, sort = T)
head(b)
```

```
##     os       n
## 1: 19 2410148
## 2: 13 2199778
## 3: 17  531695
## 4: 18  483602
## 5: 22  365576
## 6: 10  285907
```

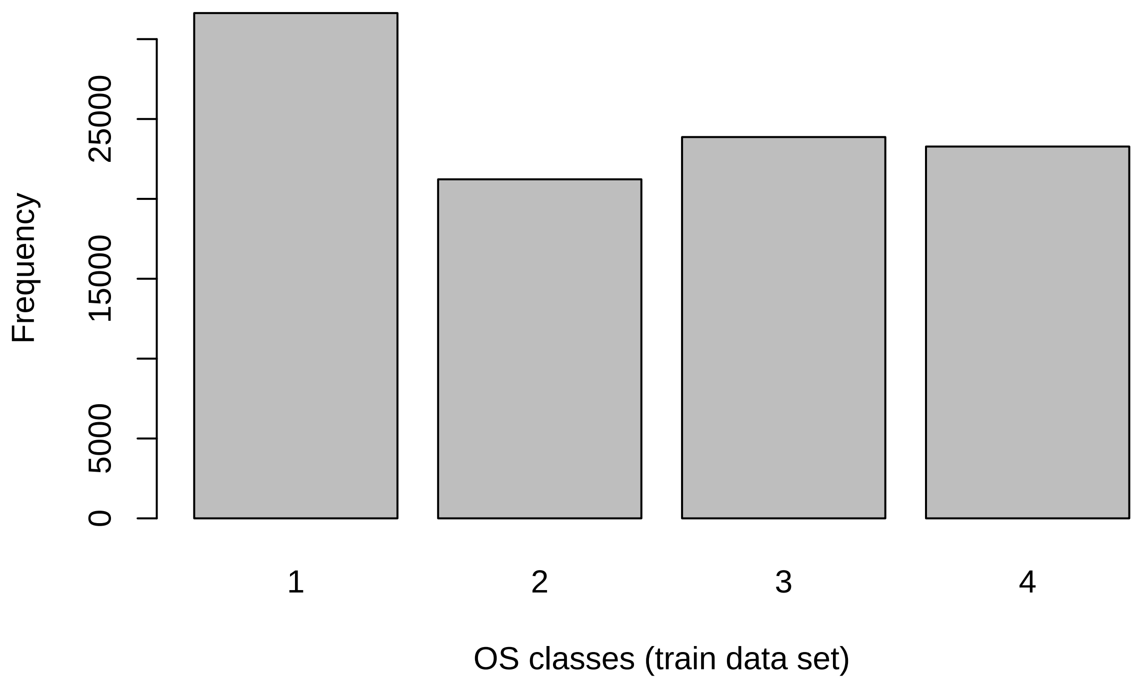
```r
# Type 19 and 13 os proportion
( (a[1,2] + a[2,2]) / sum(a) )
```

```
## [1] 0.4005952
```

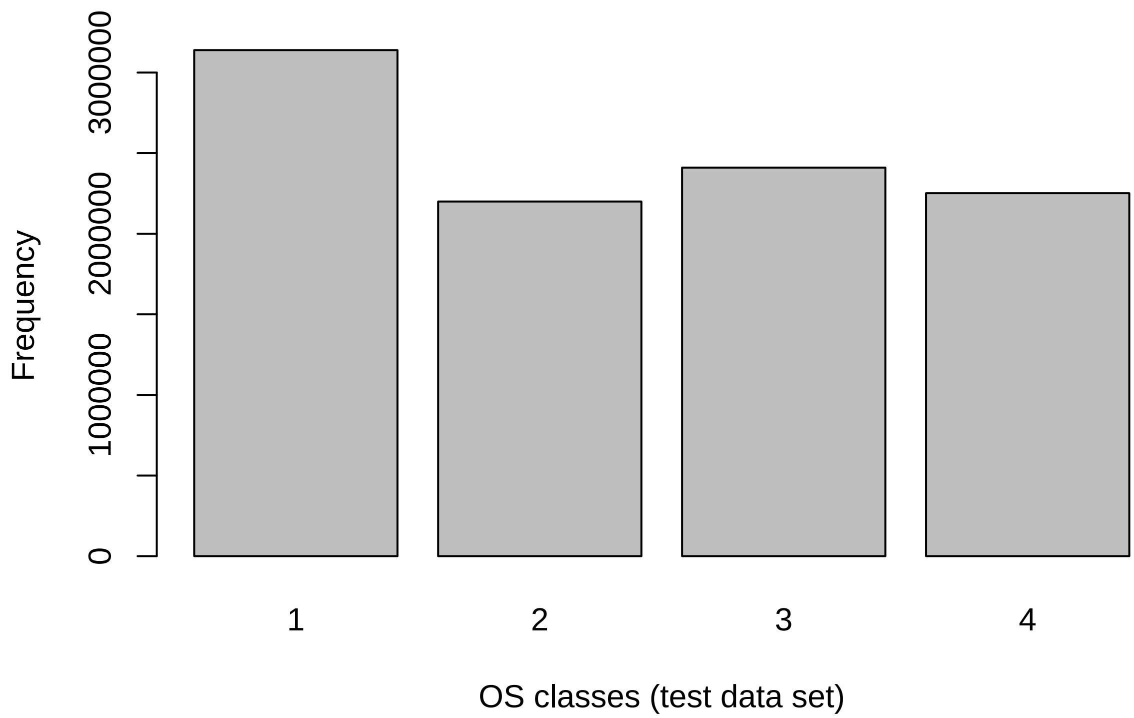
```r
( (b[1,2] + b[2,2]) / sum(b) )
```

```
##             n
## 1: 0.4577447
```

```r
                                             # Type 19 and 13 os represent almost
                                             # 40% of the systems

# Making classes for os features
class_os <- function(x) {
  if (x == 13) {2}
  else if (x == 19) {3}
  else if (x > 19) {4}
  else {1}
}

train_set$os_fac <- as.factor(sapply(train_set$os, class_os))
plot(train_set$os_fac, xlab = 'OS classes (train data set)', ylab = 'Frequency')
```




```r
test_set$os_fac <- as.factor(sapply(test_set$os, class_os))
plot(test_set$os_fac, xlab = 'OS classes (test data set)', ylab = 'Frequency')
```




```
# Channel feature
sort(unique(train_set$channel))
```

```
##   [1]   3   4   5  13  15  17  18  19  21  22  24  30 101 105 107 108 110 111
##  [19] 113 114 115 116 118 120 121 122 123 124 125 126 127 128 130 134 135 137
##  [37] 138 140 145 150 153 160 171 173 174 178 182 203 205 208 210 211 212 213
##  [55] 215 219 224 232 234 236 237 242 243 244 245 253 258 259 261 262 265 266
##  [73] 268 272 274 277 278 280 282 315 317 319 320 322 325 326 328 330 332 333
##  [91] 334 340 341 343 347 349 353 356 360 361 364 371 373 376 377 379 386 391
## [109] 400 401 402 404 406 409 410 411 412 416 417 419 420 421 424 430 435 439
## [127] 442 445 446 448 449 450 451 452 453 455 456 457 459 460 463 465 466 467
## [145] 469 474 477 478 479 480 481 483 484 486 487 488 489 490 496 497 498
```

```
sort(unique(test_set$channel))
```

```
##   [1]   0   3   4   5  13  15  17  18  19  21  22  24  30 101 105 107 108 110
##  [19] 111 113 114 115 116 118 120 121 122 123 124 125 126 128 129 130 134 135
##  [37] 137 138 140 142 145 150 153 160 171 173 174 178 181 182 203 205 208 210
##  [55] 211 212 213 215 219 222 223 224 225 232 234 236 237 238 242 243 244 245
##  [73] 251 253 258 259 261 262 265 266 268 272 274 277 278 280 281 282 311 315
##  [91] 317 319 320 325 326 328 330 332 333 334 340 341 343 347 349 352 353 356
## [109] 360 361 364 371 373 376 377 379 386 391 400 401 402 406 407 409 410 411
## [127] 412 414 416 417 419 420 421 424 430 435 439 442 445 446 449 450 451 452
## [145] 453 456 457 458 459 460 463 465 466 467 469 471 477 478 479 480 481 483
## [163] 484 486 487 488 489 496 497 498
```

```
summary(train_set$channel)
```

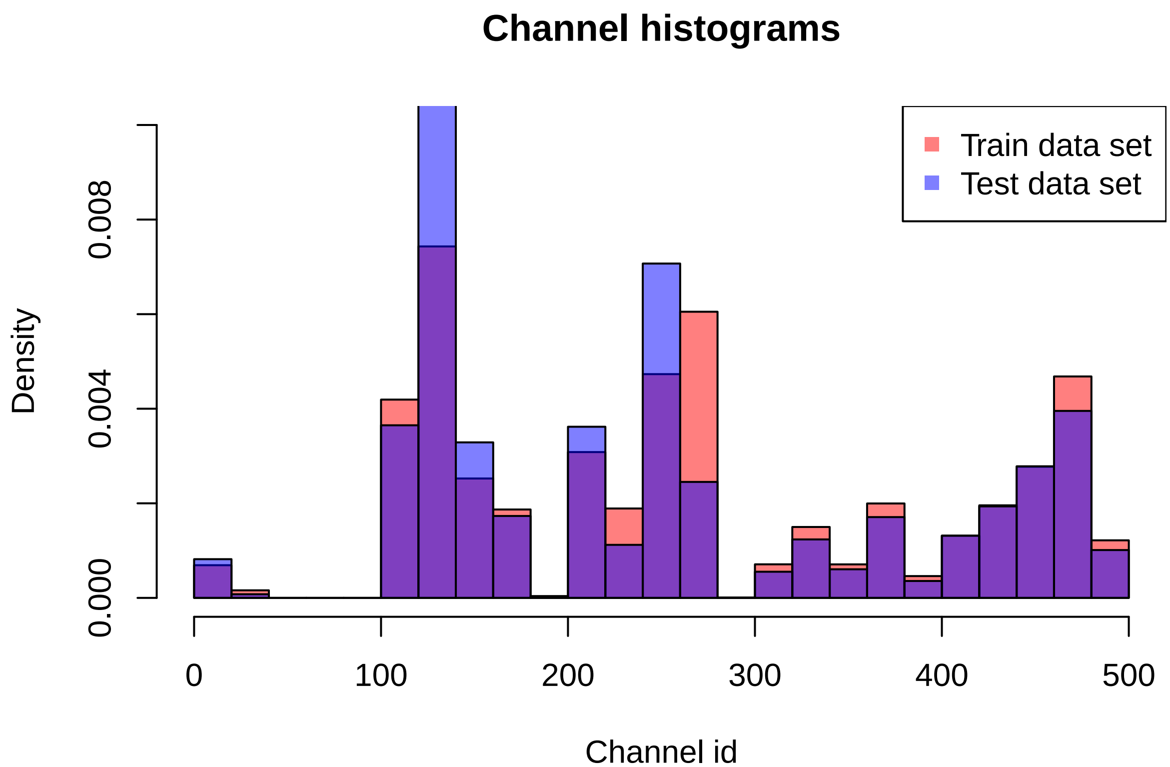
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     3.0   145.0   258.0   268.8   379.0   498.0
```

```
summary(test_set$channel)
```

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

```
##     0.0   134.0   237.0   252.7   377.0   498.0
```

```
# histograms
hist(train_set$channel, freq = F, ylim = c(0, 0.01), breaks = 20,
     col = rgb(1,0,0,0.5), main = 'Channel histograms', xlab = 'Channel id')
hist(test_set$channel, freq = F, breaks = 20,
     col = rgb(0,0,1,0.5), add = T)
legend(x = "topright", legend = c('Train data set','Test data set'),
       col = c(rgb(1,0,0,0.5), rgb(0,0,1,0.5)), pch = 15)
```




```
# Countings
a <- train_set %>%
  count(channel, sort = T)
head(a)
```

```
##   channel    n
## 1     280 8114
## 2     245 4802
## 3     107 4543
## 4     477 3960
## 5     134 3224
## 6     259 3130
```

```
b <- test_set %>%
  count(channel, sort = T)
head(b)
```

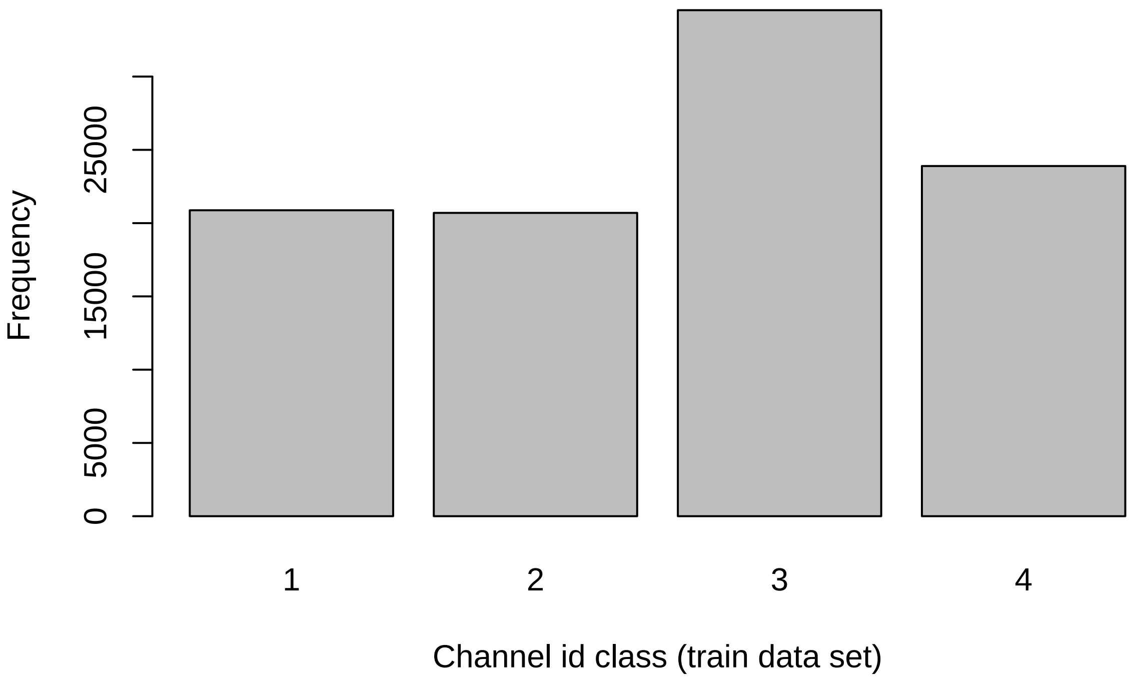
```
##    channel      n
## 1:     245 793105
## 2:     134 630888
## 3:     259 469845
## 4:     477 412559
```

```
## 5:     121 402226
## 6:     107 388035
```

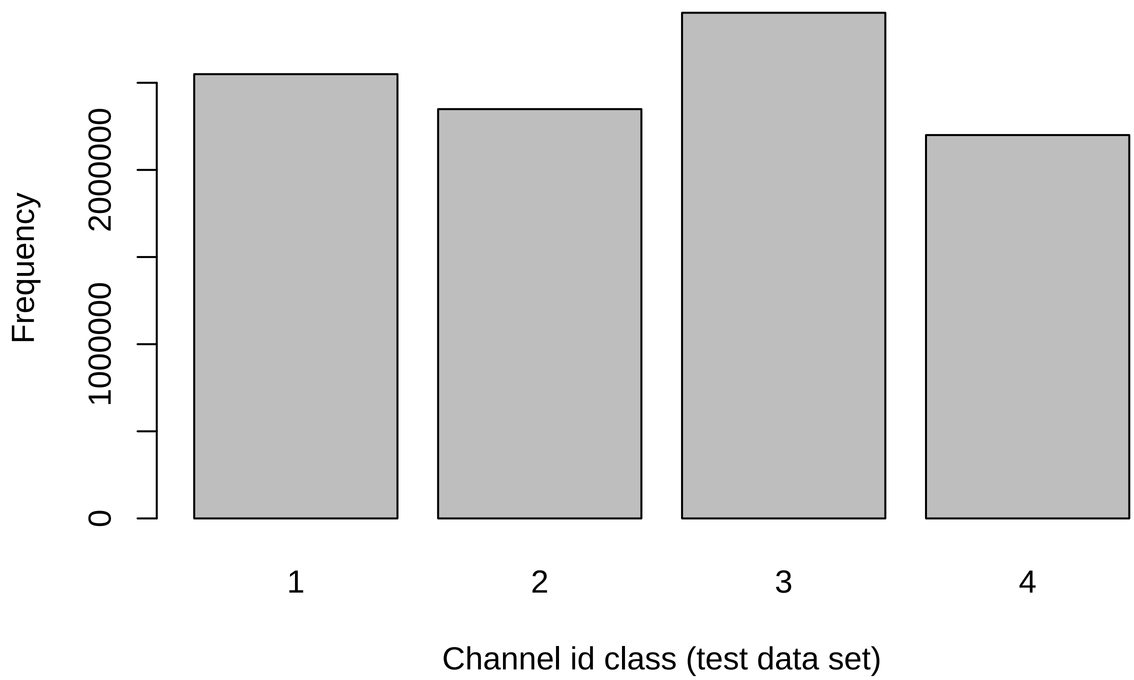
```r
#  Balancing the four channel classes
div_channel <- bin_data(c(train_set$channel, test_set$channel),
                        bins = 4, binType = "quantile")
levels(div_channel)
```

```
## [1] "[0, 134)"   "[134, 242)" "[242, 377)" "[377, 498]"
```

```r
train_set$channel_fac <- cut(train_set$channel,
                             breaks = c(0, 135, 236, 401, nrow(train_set)),
                             right = F, labels = c(1, 2, 3, 4))

test_set$channel_fac <- cut(test_set$channel,
                            breaks = c(0, 135, 236, 401, nrow(test_set)),
                            right = F, labels = c(1, 2, 3, 4))

plot(train_set$channel_fac, xlab = 'Channel id class (train data set)',
     ylab = 'Frequency')
```




```r
plot(test_set$channel_fac, xlab = 'Channel id class (test data set)',
     ylab = 'Frequency')
```




```r
# Features that does not contain missing values
dim(train_set)
```

```
## [1] 100000     18
```

```r
any(is.na(train_set[,1:6]))
```

```
## [1] FALSE
```

```r
any(is.na(train_set[,8:11]))
```

```
## [1] FALSE
```

```r
any(is.na(train_set[,14:17]))
```

```
## [1] TRUE
```

```r
# Dealing with the features with missing values
any(is.na(train_set[,7]))
```

```
## [1] TRUE
```

```r
labels(train_set)[[2]][7]
```

```
## [1] "attributed_time"
```

```r
head(unique(train_set$attributed_time))
```

```
## [1] NA                        "2017-11-08 02:22:38 UTC"
## [3] "2017-11-08 06:10:37 UTC" "2017-11-07 11:59:05 UTC"
## [5] "2017-11-09 11:52:01 UTC" "2017-11-08 01:55:02 UTC"
```

```r
                                        # This features will not be utilized

any(is.na(train_set[,12]))
```

```
## [1] FALSE
```

```r
labels(train_set)[[2]][12]
```

```
## [1] "click_day"
```

```r
unique(train_set$attributed_day)
```

```
## [1] NA  8  7  9  6
```

```r
                                        # This features will not be utilized

any(is.na(train_set[,13]))
```

```
## [1] TRUE
```

```r
labels(train_set)[[2]][13]
```

```
## [1] "attributed_day"
```

```r
unique(train_set$attributed_hour)
```

```
##  [1] NA  2  6 12 13 23  9  5 10 20  7  0  4  8 15 11  1 14 17  3 16 22 24 21
```

```r
                                        # This features will not be utilized


# Reducing the quantity of not downloaded to balance the train target feature
n <- nrow(train_set[train_set$is_attributed == 1, ])
n
```

```
## [1] 227
```

```r
train_no <- train_set %>%
  filter(is_attributed == 0) %>%
  slice_sample(n = n, replace = F)
nrow(train_no)
```

```
## [1] 227
```

```r
train_yes <- train_set %>%
  filter(is_attributed == 1)
nrow(train_yes)
```

```
## [1] 227
```

```r
train_set1 <- rbind(train_no, train_yes)
train_set1 <- train_set1 %>%
  slice_sample(n = nrow(train_set1), replace = F)
nrow(train_set1)/2
```

```
## [1] 227
```

```r
# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
ls()
```

```
## [1] "test_set"   "train_set"  "train_set1"
```

```r
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2412712 128.9   13886779  741.7  21698091 1158.9
## Vcells 90545109 690.9  235604878 1797.6 235582774 1797.4
```

```r
################################################################################
################################################################################
################################################################################

# logistic regression model
labels(test_set)[[2]]
```

```
##  [1] "ip"                "app"               "device"            "os"
##  [5] "channel"           "click_time"        "attributed_time" "is_attributed"
##  [9] "repetitions"       "repetitions_fac" "click_day"         "app_fac"
## [13] "device_fac"        "os_fac"            "channel_fac"
```

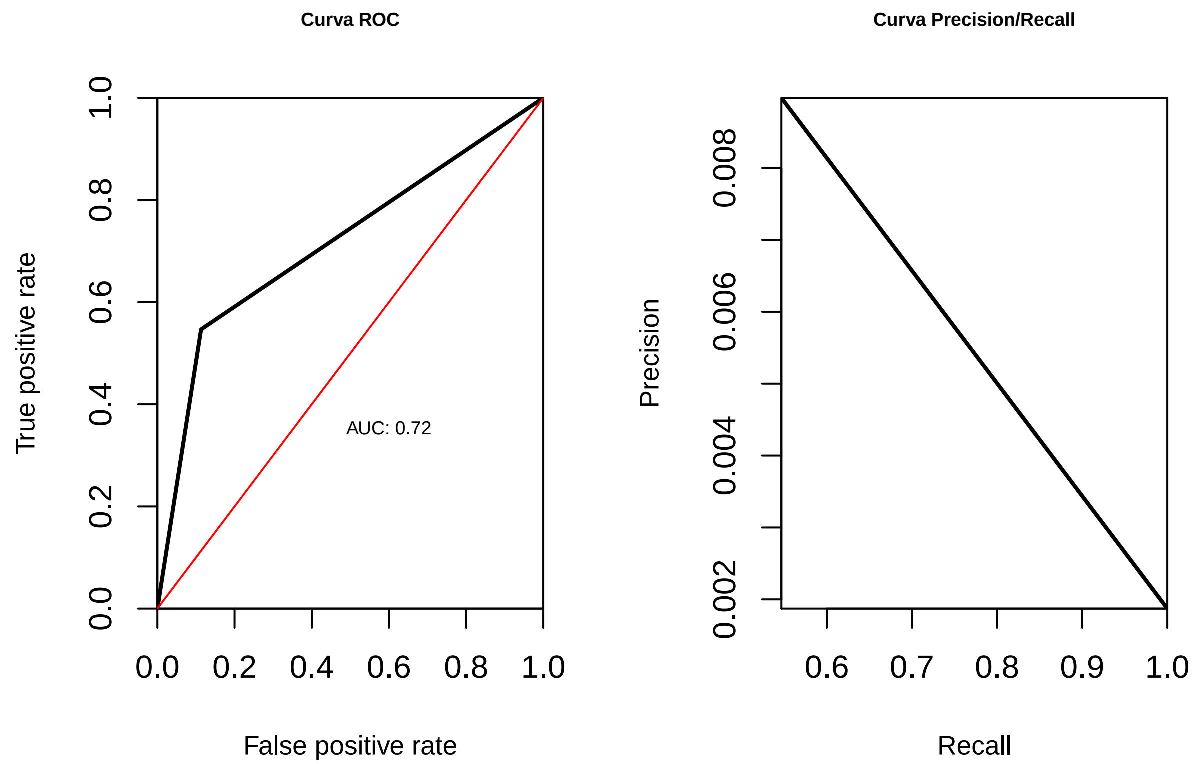
```r
model1 <- glm(is_attributed ~ repetitions_fac + app_fac +
                device_fac + os_fac + channel_fac,
                data = train_set1,
                family = "binomial")

# Summary of the model
summary(model1)
```

```
##
## Call:
## glm(formula = is_attributed ~ repetitions_fac + app_fac + device_fac +
##     os_fac + channel_fac, family = "binomial", data = train_set1)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.82377  -0.50931   0.09675   0.67433   2.86165
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -17.6449   960.2457  -0.018  0.98534
## repetitions_fac2  -1.6305     0.3241  -5.031 4.89e-07 ***
## app_fac2          17.7148   960.2456   0.018  0.98528
## app_fac3          15.6071   960.2458   0.016  0.98703
## app_fac4          19.0103   960.2456   0.020  0.98421
## device_fac2        2.1978     0.5086   4.321 1.55e-05 ***
## os_fac2            0.4329     0.4138   1.046  0.29546
## os_fac3            0.7836     0.3849   2.036  0.04179 *
## os_fac4            0.4050     0.3364   1.204  0.22871
## channel_fac2      -0.6466     0.3663  -1.765  0.07757 .
## channel_fac3      -0.8423     0.3755  -2.243  0.02490 *
## channel_fac4      -1.1879     0.3990  -2.977  0.00291 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 629.38  on 453  degrees of freedom
## Residual deviance: 360.18  on 442  degrees of freedom
## AIC: 384.18
##
## Number of Fisher Scoring iterations: 17
```

```r
# Predictions
predictions1 <- predict(model1, test_set, type="response")
predictions1 <- round(predictions1)

# Evaluation
confusionMatrix(as.factor(predictions1),
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 8851380    8485
##          1 1129903   10232
##
##                Accuracy : 0.8862
##                  95% CI : (0.886, 0.8864)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.014
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.546669
##             Specificity : 0.886798
##          Pos Pred Value : 0.008974
##          Neg Pred Value : 0.999042
##              Prevalence : 0.001872
##          Detection Rate : 0.001023
##    Detection Prevalence : 0.114014
##       Balanced Accuracy : 0.716733
##
##        'Positive' Class : 1
##
```

```r
# ROC curve
predictions1_roc <- prediction(predictions1, test_set$is_attributed)
source("plot_utils.R")
par(mfrow = c(1,2))
plot.roc.curve(predictions1_roc, title.text = "Curva ROC")
plot.pr.curve(predictions1_roc, title.text = "Curva Precision/Recall")
```




```
par(mfrow = c(1,1))

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##              used   (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells   2449468  130.9   16117742  860.8  21698091 1158.9
## Vcells  94598634  721.8  285871895 2181.1 437282775 3336.3
```

```
################################################################################
################################################################################
################################################################################
```

```
# logistic regression model with the most significant variables
model2 <- glm(is_attributed ~ repetitions + device_fac + os_fac,
              data = train_set1,
              family = "binomial")

# Summary of the model
summary(model2)
```

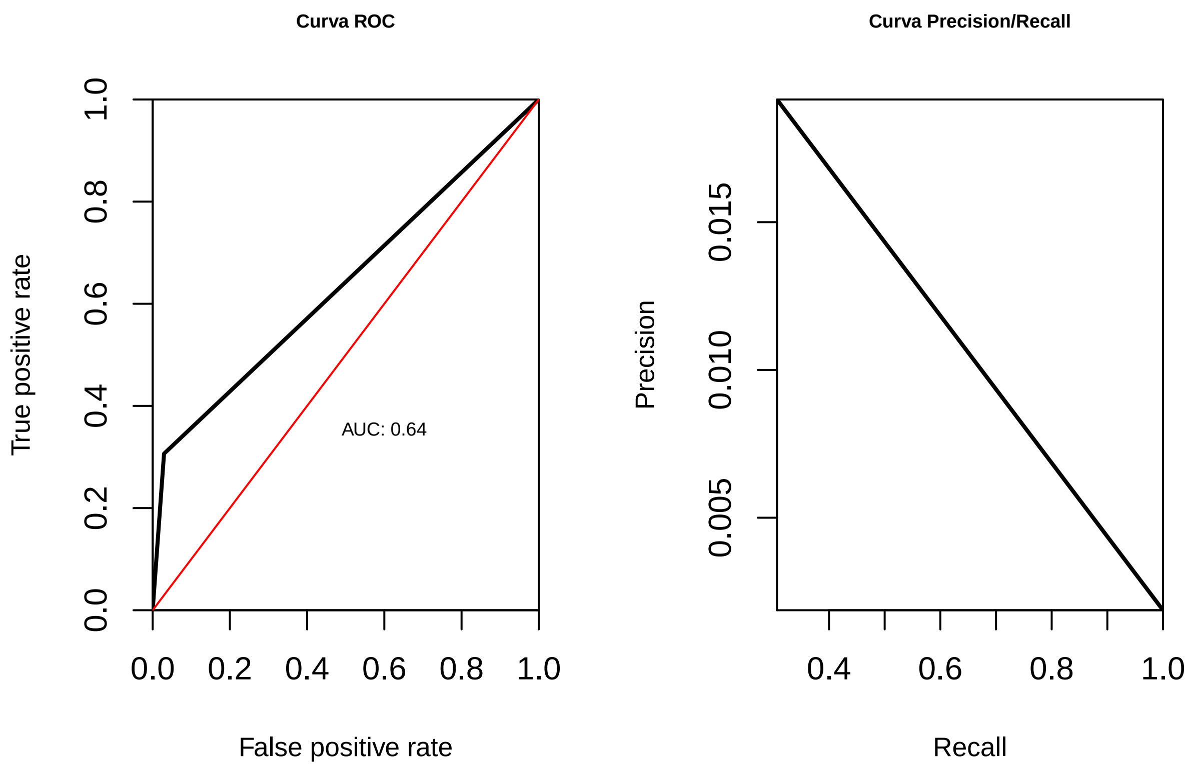
```
##
## Call:
## glm(formula = is_attributed ~ repetitions + device_fac + os_fac,
##     family = "binomial", data = train_set1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2255  -1.0340  -0.1042   1.2220   2.0650
##
```

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.713642   0.194704  -3.665 0.000247 ***
## repetitions -0.002098   0.001136  -1.847 0.064703 .
## device_fac2  2.613316   0.373279   7.001 2.54e-12 ***
## os_fac2      0.368682   0.327697   1.125 0.260560
## os_fac3      0.629404   0.290971   2.163 0.030532 *
## os_fac4      0.495354   0.261740   1.893 0.058419 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 629.38  on 453  degrees of freedom
## Residual deviance: 543.22  on 448  degrees of freedom
## AIC: 555.22
##
## Number of Fisher Scoring iterations: 4
```

```r
# Predictions
predictions2 <- predict(model2, test_set, type="response")
predictions2 <- round(predictions2)

# Evaluation
confusionMatrix(as.factor(predictions2),
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9687434   12980
##          1  293849    5737
##
##                Accuracy : 0.9693
##                  95% CI : (0.9692, 0.9694)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0326
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3065128
##             Specificity : 0.9705600
##          Pos Pred Value : 0.0191498
##          Neg Pred Value : 0.9986619
##              Prevalence : 0.0018717
##          Detection Rate : 0.0005737
##    Detection Prevalence : 0.0299586
##       Balanced Accuracy : 0.6385364
##
##        'Positive' Class : 1
##
```

```r
# Criando curvas ROC
predictions2_roc <- prediction(predictions2, test_set$is_attributed)
source("plot_utils.R")
par(mfrow = c(1,2))
plot.roc.curve(predictions2_roc, title.text = "Curva ROC")
plot.pr.curve(predictions2_roc, title.text = "Curva Precision/Recall")
```




```r
par(mfrow = c(1,1))

#  Conclusion: the AUC value decrease in relation to the previous model.
#              The AUC value is the Balanced Accuracy of the
#              confusionMatrix results.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2449465 130.9   14253705  761.3  21698091 1158.9
## Vcells 94599086 721.8  285871895 2181.1 437282775 3336.3
```

```r
detach(package:ROCR)

################################################################################
################################################################################
################################################################################
```

```r
# KSVM model with rbf kernel
library(kernlab)
```

```
##
```

```
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
model3 <- ksvm(is_attributed ~ repetitions + app_fac +
                 device_fac + os_fac + channel_fac,
               data = train_set1,
               kernel = 'rbf')

# Summary of the model
summary(model3)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```
# Predictions
predictions3 <- predict(model3, test_set, type="response")

# Evaluation
confusionMatrix(predictions3,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9323962    8431
##          1  657321   10286
##
##                Accuracy : 0.9334
##                  95% CI : (0.9333, 0.9336)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0264
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.549554
##             Specificity : 0.934145
##          Pos Pred Value : 0.015407
##          Neg Pred Value : 0.999097
##              Prevalence : 0.001872
##          Detection Rate : 0.001029
##    Detection Prevalence : 0.066761
##       Balanced Accuracy : 0.741849
##
##        'Positive' Class : 1
##
```

```
# Conclusion: the first model is still the best one.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
```

```r
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2739734 146.4   16331636  872.3  21698091 1158.9
## Vcells 94918774 724.2  465549826 3551.9 581934800 4439.9
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# KSVM model with rbf kernel and the most significant variables
model4 <- ksvm(is_attributed ~ repetitions + device_fac + os_fac,
               data = train_set1,
               kernel = 'rbf')

# Predictions
predictions4 <- predict(model4, test_set, type="response")

# Evaluation
confusionMatrix(predictions4,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9788661   13171
##          1  192622    5546
##
##                Accuracy : 0.9794
##                  95% CI : (0.9793, 0.9795)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0479
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.2963082
##             Specificity : 0.9807017
##          Pos Pred Value : 0.0279864
##          Neg Pred Value : 0.9986563
##              Prevalence : 0.0018717
##          Detection Rate : 0.0005546
##    Detection Prevalence : 0.0198168
##       Balanced Accuracy : 0.6385049
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: the first model is still the best one. It is worst than the
#              previous model.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
```

```r
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2741669 146.5   13065309  697.8  21698091 1158.9
## Vcells 94941047 724.4  372439861 2841.5 581934800 4439.9
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# KSVM model with vanilladot Linear kernel
model5 <- ksvm(is_attributed ~ repetitions + app_fac +
                 device_fac + os_fac + channel_fac,
               data = train_set1,
               kernel = 'vanilla')
```

```
##  Setting default kernel parameters
```

```r
# Predictions
predictions5 <- predict(model5, test_set, type="response")

# Evaluation
confusionMatrix(predictions5,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 7683916    4700
##          1 2297367   14017
##
##                Accuracy : 0.7698
##                  95% CI : (0.7695, 0.7701)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0083
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.748891
##             Specificity : 0.769832
##          Pos Pred Value : 0.006064
##          Neg Pred Value : 0.999389
##              Prevalence : 0.001872
##          Detection Rate : 0.001402
##    Detection Prevalence : 0.231138
##       Balanced Accuracy : 0.759362
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: now this is the best model so far.

# Cleaning the house
```

```r
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2741960 146.5   16903854  902.8  21698091 1158.9
## Vcells 94941793 724.4  429191520 3274.5 581934800 4439.9
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# KSVM model with vanilladot Linear kernel and the most significant variables
model6 <- ksvm(is_attributed ~ repetitions + device_fac + os_fac,
               data = train_set1,
               kernel = 'vanilla')
```

```
##  Setting default kernel parameters
```

```r
# Predictions
predictions6 <- predict(model6, test_set, type="response")

# Evaluation
confusionMatrix(predictions6,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9368709   12437
##          1  612574    6280
##
##                Accuracy : 0.9375
##                  95% CI : (0.9373, 0.9376)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0161
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.335524
##             Specificity : 0.938628
##          Pos Pred Value : 0.010148
##          Neg Pred Value : 0.998674
##              Prevalence : 0.001872
##          Detection Rate : 0.000628
##    Detection Prevalence : 0.061885
##       Balanced Accuracy : 0.637076
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: the model 5 scores better.

# Cleaning the house
```

```r
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2741975 146.5   13523084  722.3  21698091 1158.9
## Vcells 94942100 724.4  343353216 2619.6 581934800 4439.9
```

```r
detach(package:kernlab)

################################################################################
################################################################################
################################################################################
```

```r
# SVM model with radial kernel
library(e1071)
```

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:mltools':
##
##     skewness
```

```r
model7 <- svm(is_attributed ~ repetitions + app_fac +
                 device_fac + os_fac + channel_fac,
               data = train_set1,
               kernel = 'radial')

# Predictions
predictions7 <- predict(model7, test_set, type="response")

# Evaluation
confusionMatrix(predictions7,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 7854572    6187
##          1 2126711   12530
##
##                Accuracy : 0.7867
##                  95% CI : (0.7865, 0.787)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0079
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.669445
##             Specificity : 0.786930
##          Pos Pred Value : 0.005857
##          Neg Pred Value : 0.999213
##              Prevalence : 0.001872
```

```
##          Detection Rate : 0.001253
##    Detection Prevalence : 0.213924
##       Balanced Accuracy : 0.728187
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: the model 5 scores better.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2747045 146.8   15921902  850.4  21698091 1158.9
## Vcells 94954745 724.5  388693684 2965.5 599975828 4577.5
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# SVM model with radial kernel and the most significant variables
model8 <- svm(is_attributed ~ repetitions + device_fac + os_fac,
              data = train_set1,
              kernel = 'radial')

# Predictions
predictions8 <- predict(model8, test_set, type="response")

# Evaluation
confusionMatrix(predictions8,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 8297905   12464
##          1 1683378    6253
##
##                Accuracy : 0.8304
##                  95% CI : (0.8302, 0.8306)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0036
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.3340813
##             Specificity : 0.8313465
##          Pos Pred Value : 0.0037008
##          Neg Pred Value : 0.9985002
##              Prevalence : 0.0018717
##          Detection Rate : 0.0006253
##    Detection Prevalence : 0.1689631
```

```
##       Balanced Accuracy : 0.5827139
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: this model is not good.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2747015 146.8   14567556  778.0  21698091 1158.9
## Vcells 94954915 724.5  298567949 2277.9 599975828 4577.5
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# SVM model with linear kernel
model9 <- svm(is_attributed ~ repetitions + app_fac +
                device_fac + os_fac + channel_fac,
              data = train_set1,
              kernel = 'linear',
              type = 'C-classification')

# Predictions
predictions9 <- predict(model9, test_set, type="response")

# Evaluation
confusionMatrix(predictions9,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 7683916    4700
##          1 2297367   14017
##
##                Accuracy : 0.7698
##                  95% CI : (0.7695, 0.7701)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0083
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.748891
##             Specificity : 0.769832
##          Pos Pred Value : 0.006064
##          Neg Pred Value : 0.999389
##              Prevalence : 0.001872
##          Detection Rate : 0.001402
##    Detection Prevalence : 0.231138
```

```
##        Balanced Accuracy : 0.759362
##
##         'Positive' Class : 1
##
```

```r
# Conclusion: it is equal to model 5.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##             used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2747032 146.8   14567687  778.0  21698091 1158.9
## Vcells 94955259 724.5  396505560 3025.1 599975828 4577.5
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# SVM model with linear kernel and the most significant variables
model10 <- svm(is_attributed ~ repetitions + device_fac + os_fac,
              data = train_set1,
              kernel = 'linear',
              type = 'C-classification')

# Predictions
predictions10 <- predict(model10, test_set, type="response")

# Evaluation
confusionMatrix(predictions10,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9368709   12437
##          1  612574    6280
##
##                Accuracy : 0.9375
##                  95% CI : (0.9373, 0.9376)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0161
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.335524
##             Specificity : 0.938628
##          Pos Pred Value : 0.010148
##          Neg Pred Value : 0.998674
##              Prevalence : 0.001872
##          Detection Rate : 0.000628
##    Detection Prevalence : 0.061885
##       Balanced Accuracy : 0.637076
```

```
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: the model 5 scores better.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2747047 146.8   14567646  778.0  21698091 1158.9
## Vcells 94955564 724.5  304567471 2323.7 599975828 4577.5
```

```r
detach(package:e1071)

################################################################################
################################################################################
################################################################################
```

```r
# Regression Trees model
library(rpart.plot)
```

```
## Loading required package: rpart
```

```r
model11 <- rpart(is_attributed ~ repetitions + app_fac +
                   device_fac + os_fac + channel_fac,
                 data = train_set1)

# Predictions
predictions11 <- predict(model11, test_set, type="class")

# Evaluation
confusionMatrix(predictions11,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 8904014    6308
##          1 1077269   12409
##
##                Accuracy : 0.8916
##                  95% CI : (0.8914, 0.8918)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0188
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.662980
##             Specificity : 0.892071
##          Pos Pred Value : 0.011388
##          Neg Pred Value : 0.999292
##              Prevalence : 0.001872
```

```
##          Detection Rate : 0.001241
##    Detection Prevalence : 0.108968
##       Balanced Accuracy : 0.777526
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: it is the best model so far.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2757720 147.3   11654117  622.4  21698091 1158.9
## Vcells 94973020 724.6  304567471 2323.7 599975828 4577.5
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# Evaluation of the most important features for the model
model12 <- train(is_attributed ~ repetitions + app_fac +
                   device_fac + os_fac + channel_fac,
                 data = train_set1,
                 method = 'rpart')
varImp(model12)
```

```
## rpart variable importance
##
##              Overall
## app_fac4     100.000
## repetitions   98.529
## app_fac3      55.844
## device_fac2   46.457
## app_fac2      31.911
## channel_fac4  17.948
## os_fac4       15.882
## channel_fac3  11.223
## channel_fac2   4.996
## os_fac2        0.000
## os_fac3        0.000
```

```r
# Regression Trees model with the most significant variables
model12 <- rpart(is_attributed ~ repetitions + app_fac +
                   device_fac + channel_fac,
                 data = train_set1)

# Predictions
predictions12 <- predict(model12, test_set, type="class")

# Evaluation
confusionMatrix(predictions12,
                reference = test_set$is_attributed, positive = '1')
```

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

```
##           Reference
## Prediction       0       1
##          0 9250892    7180
##          1  730391   11537
##
##                Accuracy : 0.9262
##                  95% CI : (0.9261, 0.9264)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0268
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.616392
##             Specificity : 0.926824
##          Pos Pred Value : 0.015550
##          Neg Pred Value : 0.999224
##              Prevalence : 0.001872
##          Detection Rate : 0.001154
##    Detection Prevalence : 0.074193
##       Balanced Accuracy : 0.771608
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: the model 11 is still the best model.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2800540 149.6   14690047  784.6  21698091 1158.9
## Vcells 95078269 725.4  320115343 2442.3 599975828 4577.5
```

```r
detach(package:rpart.plot)

################################################################################
################################################################################
################################################################################
```

```r
# Another Regression Trees model
library(C50)
model13 <- C5.0(is_attributed ~ repetitions_fac + app_fac +
                  device_fac + os_fac + channel_fac,
                data = train_set1,
                trials = 10,
                cost = matrix(c(0, 8, 1, 0), nrow = 2,
                              dimnames = list(c('0','1'), c('0', '1'))))

# Predictions
predictions13 <- predict(model13, test_set, type="class")

# Evaluation
confusionMatrix(predictions13,
```

```r
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 7755757    4474
##          1 2225526   14243
##
##                Accuracy : 0.777
##                  95% CI : (0.7767, 0.7773)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0089
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.760966
##             Specificity : 0.777030
##          Pos Pred Value : 0.006359
##          Neg Pred Value : 0.999423
##              Prevalence : 0.001872
##          Detection Rate : 0.001424
##    Detection Prevalence : 0.223977
##       Balanced Accuracy : 0.768998
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: it is the best model so far.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2814109 150.3   17906369  956.4  22382961 1195.4
## Vcells 95101264 725.6  371195240 2832.0 599975828 4577.5
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# Another Regression Trees model with the most significant variables
model14 <- C5.0(is_attributed ~ repetitions + app_fac +
                  device_fac + channel_fac,
                data = train_set1,
                trials = 10,
                cost = matrix(c(0, 2, 1, 0), nrow = 2,
                              dimnames = list(c('0','1'), c('0', '1'))))

# Predictions
predictions14 <- predict(model14, test_set, type="class")
```

```r
# Evaluation
confusionMatrix(predictions14,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 7685419    4598
##          1 2295864   14119
##
##                Accuracy : 0.77
##                  95% CI : (0.7697, 0.7702)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0084
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.754341
##             Specificity : 0.769983
##          Pos Pred Value : 0.006112
##          Neg Pred Value : 0.999402
##              Prevalence : 0.001872
##          Detection Rate : 0.001412
##    Detection Prevalence : 0.230998
##       Balanced Accuracy : 0.762162
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: The previous model was better.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2814070 150.3   14325096  765.1  22382961 1195.4
## Vcells 95101553 725.6  296956192 2265.6 599975828 4577.5
```

```r
detach(package:C50)

################################################################################
################################################################################
################################################################################
```

```r
# Random Forest model
library(randomForest)
```

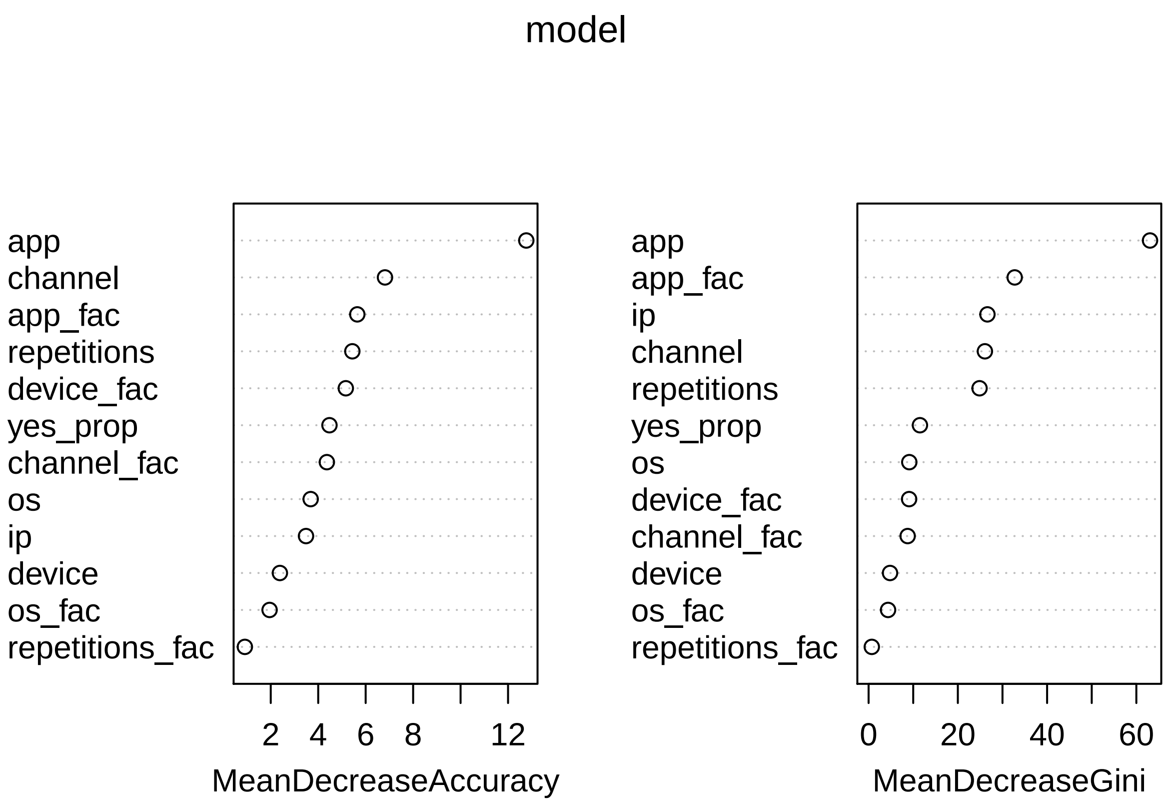
```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(ggplot2)
# Feature importances
model <- randomForest(is_attributed ~ ip + app + device + os + channel +
                        repetitions + yes_prop + repetitions_fac + app_fac +
                        device_fac + os_fac + channel_fac,
                      data = train_set1,
                      ntree = 30,
                      nodesize = 1, importance = T)

varImpPlot(model)
```




```r
# Random forest model
model15 <- randomForest(is_attributed ~ repetitions_fac * app +
                          channel * app_fac,
                data = train_set1,
                ntree = 30,
                nodesize = 1)

# Predictions
predictions15 <- predict(model15, test_set, type="class")

# Evaluation
```

```r
confusionMatrix(predictions15,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9724700    4873
##          1  256583   13844
##
##                Accuracy : 0.9739
##                  95% CI : (0.9738, 0.974)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0926
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.739648
##             Specificity : 0.974294
##          Pos Pred Value : 0.051193
##          Neg Pred Value : 0.999499
##              Prevalence : 0.001872
##          Detection Rate : 0.001384
##    Detection Prevalence : 0.027043
##       Balanced Accuracy : 0.856971
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: This is the best model.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'train_set1', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2831563 151.3   15129224  808.0  22382961 1195.4
## Vcells 95143866 725.9  432245791 3297.8 599975828 4577.5
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# Reducing the quantity of not downloaded to balance the train target feature
train_set1 <- downSample(x = train_set %>% select(-is_attributed),
                         y = train_set$is_attributed, yname = 'is_attributed')
table(train_set1$is_attributed)
```

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

```r
# Random forest model
model15 <- randomForest(is_attributed ~ repetitions_fac * app +
```

```r
                          channel * app_fac,
                        data = train_set1,
                        ntree = 30,
                        nodesize = 1)

# Predictions
predictions15 <- predict(model15, test_set, type="class")

# Evaluation
confusionMatrix(predictions15,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9860093    7384
##          1  121190   11333
##
##                Accuracy : 0.9871
##                  95% CI : (0.9871, 0.9872)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1471
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.605492
##             Specificity : 0.987858
##          Pos Pred Value : 0.085517
##          Neg Pred Value : 0.999252
##              Prevalence : 0.001872
##          Detection Rate : 0.001133
##    Detection Prevalence : 0.013252
##       Balanced Accuracy : 0.796675
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: Reducing the major target class by the downSample method did not
#             change the results.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2837841 151.6   14678647  784.0  22382961 1195.4
## Vcells 95150506 726.0  415019960 3166.4 599975828 4577.5
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# Increasing minor target class
train_set1 <- upSample(x = train_set %>% select(-is_attributed),
                         y = train_set$is_attributed, yname = 'is_attributed')
table(train_set1$is_attributed)
```

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

```r
# Random forest model
model15 <- randomForest(is_attributed ~ repetitions_fac * app +
                          channel * app_fac,
                        data = train_set1,
                        ntree = 30,
                        nodesize = 1)

# Predictions
predictions15 <- predict(model15, test_set, type="class")

# Evaluation
confusionMatrix(predictions15,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9811396    7678
##          1  169887   11039
##
##                Accuracy : 0.9822
##                  95% CI : (0.9822, 0.9823)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1076
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.589785
##             Specificity : 0.982979
##          Pos Pred Value : 0.061014
##          Neg Pred Value : 0.999218
##              Prevalence : 0.001872
##          Detection Rate : 0.001104
##    Detection Prevalence : 0.018093
##       Balanced Accuracy : 0.786382
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: Enlarging the minor target class make the results worst.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'test_set')))
```

```r
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2838286 151.6   14672556  783.6  22382961 1195.4
## Vcells 95151685 726.0  398483162 3040.2 599975828 4577.5
```

```r
################################################################################
################################################################################
################################################################################
```

```r
# Balancing the target class
library(DMwR)
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
train_set1 <- train_set %>%
  select(is_attributed, repetitions_fac, app, channel, app_fac)

train_set1 <- SMOTE(is_attributed ~ ., data  = train_set1)
table(train_set1$is_attributed)
```

```
##
##   0   1
## 908 681
```

```r
# Random forest model
model15 <- randomForest(is_attributed ~ repetitions_fac * app +
                          channel * app_fac,
                        data = train_set1,
                        ntree = 30,
                        nodesize = 1)

# Predictions
predictions15 <- predict(model15, test_set, type="class")

# Evaluation
confusionMatrix(predictions15,
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9875942    5240
##          1  105341   13477
##
##                Accuracy : 0.9889
##                  95% CI : (0.9889, 0.989)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1934
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.720041
##             Specificity : 0.989446
##          Pos Pred Value : 0.113426
##          Neg Pred Value : 0.999470
##              Prevalence : 0.001872
##          Detection Rate : 0.001348
##    Detection Prevalence : 0.011882
##       Balanced Accuracy : 0.854743
##
##        'Positive' Class : 1
##
```

```
# Conclusion: Balancing the data with SMOTE improved slightly the results.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2969708 158.6   14972388  799.7  22382961 1195.4
## Vcells 95380418 727.7  470914967 3592.8 599975828 4577.5
```

```
detach(package:DMwR)

################################################################################
################################################################################
################################################################################
```

```
# Balancing the target class
library(ROSE)
```

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

```
train_set1 <- train_set %>%
  select(is_attributed, repetitions_fac, app, channel, app_fac)

train_set1 <- ROSE(is_attributed ~ ., data  = train_set1)$data
table(train_set1$is_attributed)
```

```
##
##     0     1
## 50165 49835
```

```
# Random forest model
model15 <- randomForest(is_attributed ~ repetitions_fac * app +
                          channel * app_fac,
                        data = train_set1,
                        ntree = 30,
                        nodesize = 1)

# Predictions
predictions15 <- predict(model15, test_set, type="class")

# Evaluation
confusionMatrix(predictions15,
```

```r
                reference = test_set$is_attributed, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 9920069    7083
##          1   61214   11634
##
##                Accuracy : 0.9932
##                  95% CI : (0.9931, 0.9932)
##     No Information Rate : 0.9981
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2519
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.621574
##             Specificity : 0.993867
##          Pos Pred Value : 0.159702
##          Neg Pred Value : 0.999287
##              Prevalence : 0.001872
##          Detection Rate : 0.001163
##    Detection Prevalence : 0.007285
##       Balanced Accuracy : 0.807721
##
##        'Positive' Class : 1
##
```

```r
# Conclusion: This worse the results.

# Cleaning the house
rm(list = setdiff(ls(), c('train_set', 'test_set')))
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2972765 158.8   14829312  792.0  22382961 1195.4
## Vcells 95387268 727.8  376731974 2874.3 599975828 4577.5
```

```r
################################################################################
```

Continue on part two,

filename project_click_fraud_2_3_4_markdown_final_model.pdf