

TalkingData AdTracking Fraud Detection Challenge

Marcone

2020-06-28

```
setwd("C:/CursoDSA/BigDataAzure/Projetos")  
set.seed(42)
```

```
# Import necessary Libraries  
library(e1071)
```

```
require(lubridate)
```

```
## Loading required package: lubridate
```

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

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

```
library(readr)  
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
```

```
library(plyr)
```

```
##  
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':  
##  
##   arrange, count, desc, failwith, id, mutate, rename, summarise,  
##   summarize
```

```
library(caret)
```

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

```
## Loading required package: ggplot2
```

```
library(ROCR)
```

```
library(ggplot2)  
library(gridExtra)
```

```
##  
## Attaching package: 'gridExtra'
```

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

```
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:gridExtra':  
##  
##   combine
```

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

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

```
library(e1071)
```

```
# Read .csv data on a zipped folder
folder <- "C:/CursoDSA/BigDataAzure/Projetos/dataset_project1.zip"
con <- unz(folder,"dataset_project1/train_sample.csv")
train <- read_csv(con)
```

```
## 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()
## )
```

```
View(head(train))
```

```
# Convert time to seconds
train$click_time <- as.numeric(train$click_time)

# Check if there is NA values on each colum
for(i in 1:length(train)){if(any(is.na(train[, i]))){print(i)}}
```

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

```

## only the column 7

# Drop attributed time column as it is not going to be used
train <- train[,-7]

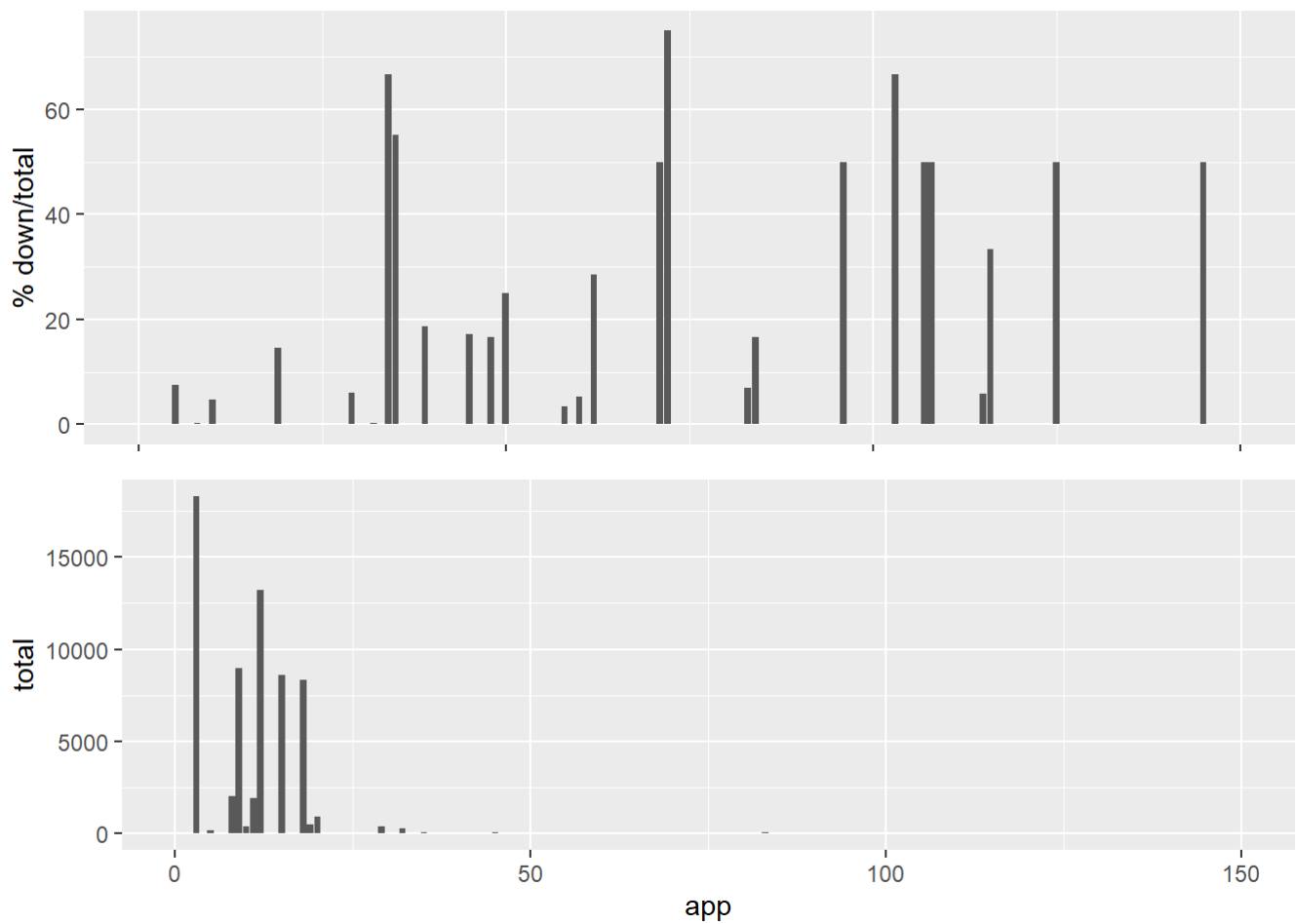
# Transform column to factor data type
train$is_attributed <- as.factor(train$is_attributed)

##### DATA EXPLORATION #####

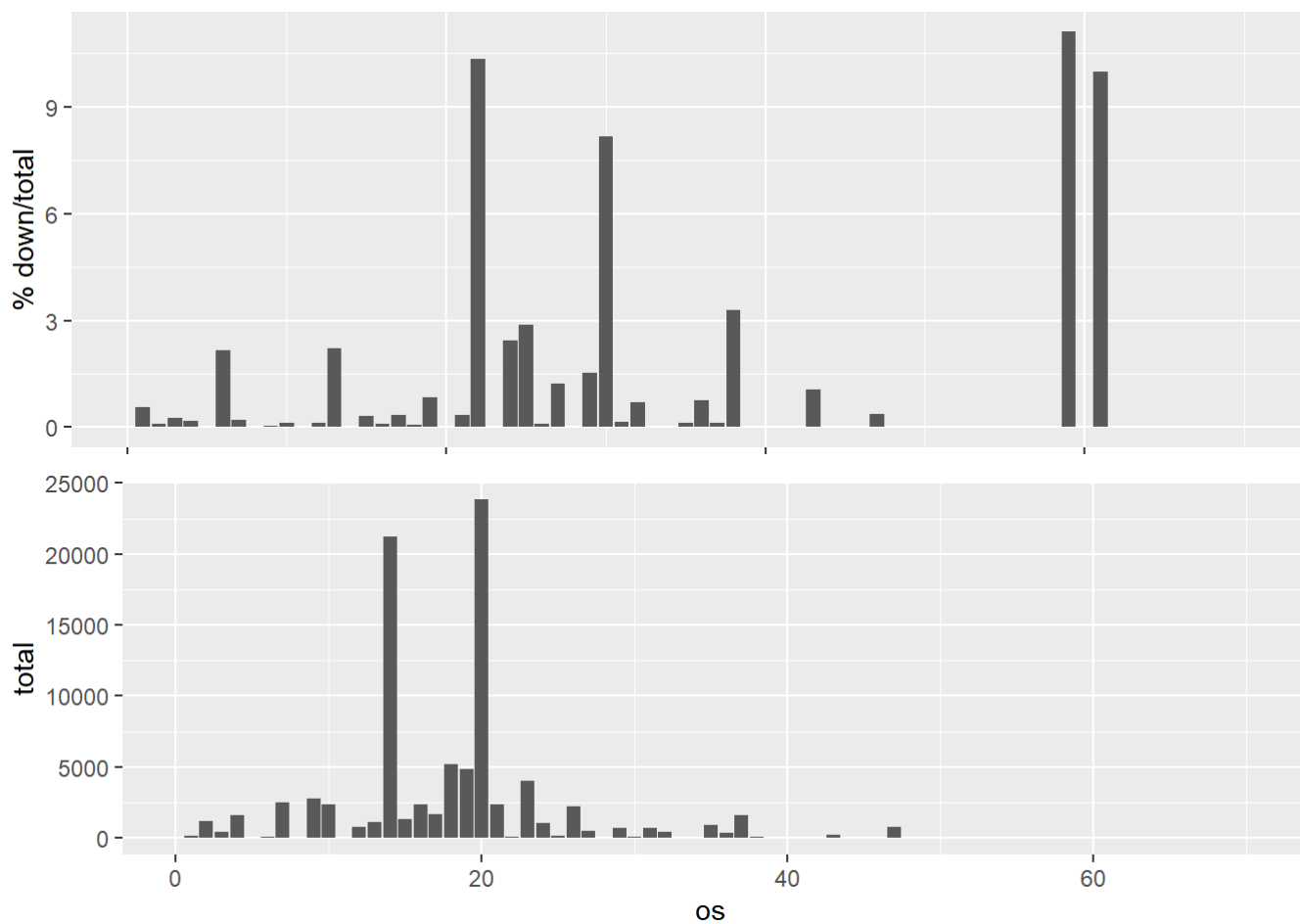
# Function that plot 2 graphs: download rate by the total of clicks (upper graph), and total
  clicks (bellow graph)
plot_graph <- function(feature,limit_n){
  item <- train%>%filter(is_attributed==1)%>%group_by_at(feature)%>%group_keys()
  df1 <- data.frame(item=item[[1]] ,
                    downloads=train%>%filter(is_attributed==1)%>%group_by_at(feature)%>%group_s
size())
  item <- train%>%filter(is_attributed==0)%>%group_by_at(feature)%>%group_keys()
  df2 <- data.frame(item=item[[1]],
                    no.downloads=train%>%filter(is_attributed==0)%>%group_by_at(feature)%>%grou
p_size())
  df <- merge.data.frame(df1, df2, by.y=0)
  df <- df%>%mutate(total=no.downloads+downloads)%>%mutate(rate=100*downloads/total)
  p1 <- ggplot(df,aes(x=item, y=rate)) + geom_bar(stat="identity") + xlim(0, limit_n) + ylab(
"% down/total") + theme(axis.title.x=element_blank(),

axis.text.x=element_blank())
  p2 <- ggplot(df,aes(x=item, y=total)) + geom_bar(stat="identity") + xlim(0, limit_n) + xlab
(feature)
  grid.arrange(p1, p2, ncol = 1)
}
# Plot function for app
plot_graph("app",150)

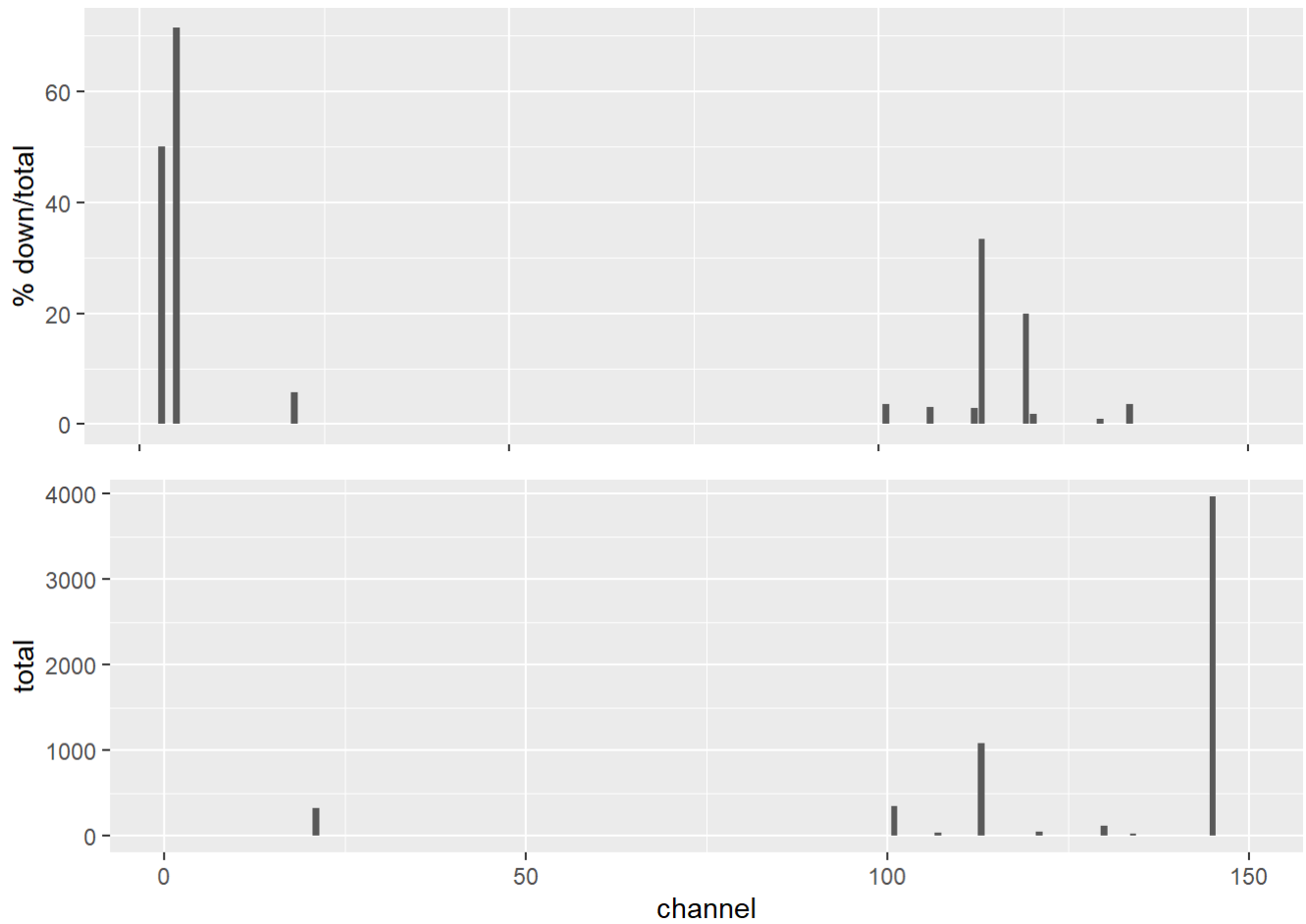
```



```
# Plot function for os
plot_graph("os",70)
```



```
# Plot function for channel
plot_graph("channel",150)
```



```
# Labels proportion
prop.table(table(train$is_attributed))
```

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

```
##### FEATURE ENGINEERING #####
```

```
variables = c("ip","app","device","os","channel")
```

```
# create colums with the time since the previous click by groups
```

```
for(variable in variables){  
  train <- arrange_at(train,c(variable,"click_time"))  
  train[[paste("delay_last_",variable,sep="")] <- c(0,apply(seq(2,length(train$click_time)),  
    function(i){ifelse(train$ip[i]==train$ip[i-1],train$click_time[i]-train$click_time[i-1],0)}))  
}
```

```
# create colums with the runned time since the first and last click by groups
```

```
for(variable in variables){  
  train <- train %>% group_by_at(variable) %>% mutate(time_first = min(click_time))  
  train[[paste("delay_first_",variable,sep="")] <- train$click_time - train$time_first  
}
```

```
# Drop time_first colum
```

```
train <- train%>%subset(select=-time_first)
```

```
##### NORMALIZATION #####
```

```
train[-7] <- sapply(train%>%select(-is_attributed), function(x){return ((x - min(x)) / (max(x) - min(x)))})  
View(head(train))
```

```
##### FEATURE SELECTION #####
```

```
train$is_attributed <- as.numeric(train$is_attributed)  
control <- rfeControl(functions = rfFuncs, method = "cv",  
  verbose = FALSE, returnResamp = "all",  
  number = 6)  
results.rfe <- rfe(x = train%>%subset(select=-is_attributed),  
  y = as.matrix(train%>%subset(select=is_attributed)),  
  sizes = 1:10,  
  rfeControl = control)
```

```
results.rfe
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (6 fold)
##
## Resampling performance over subset size:
##
## Variables      RMSE Rsquared      MAE  RMSESD RsquaredSD      MAESD Selected
##           1 0.04373   0.1570 0.003737 0.004039   0.04153 0.0003279
##           2 0.04192   0.2234 0.003490 0.003962   0.02788 0.0003433
##           3 0.04254   0.2049 0.003727 0.003858   0.02056 0.0003100
##           4 0.04277   0.1988 0.003772 0.003629   0.04046 0.0003018
##           5 0.04338   0.1737 0.003828 0.003542   0.04143 0.0003077
##           6 0.04400   0.1569 0.003621 0.003355   0.04460 0.0003309
##           7 0.04440   0.1412 0.003708 0.003579   0.04833 0.0003266
##           8 0.04474   0.1266 0.003780 0.003506   0.04326 0.0003133
##           9 0.04469   0.1377 0.003685 0.003401   0.04211 0.0003163
##          10 0.04410   0.1474 0.003709 0.003289   0.02759 0.0002954
##          16 0.04172   0.2296 0.003578 0.002230   0.05287 0.0002060      *
##
## The top 5 variables (out of 16):
##   app, channel, device, delay_first_app, delay_first_os
```

```
varImp((results.rfe))
```

```
##              Overall
## app           40.860564
## channel       37.992701
## device        23.532686
## delay_first_ip 20.769616
## delay_first_app 20.693777
## delay_first_os 20.588133
## delay_first_channel 20.470971
## delay_first_device 20.467030
## click_time    20.121574
## os            18.732756
## delay_last_ip  13.277632
## ip             8.814067
## delay_last_os   6.913896
## delay_last_app  0.000000
## delay_last_channel 0.000000
## delay_last_device 0.000000
```



```

# Select 10 top features
train_sel <- train%>%ungroup()%>%select(-c(delay_last_device,delay_last_channel,delay_last_ap
p,
                                delay_last_os,ip,delay_last_ip))
View(head(train_sel))

##### TRAINING #####

train_sel$is_attributed <- as.factor(train_sel$is_attributed)

# Split data
index <- createDataPartition(train_sel$is_attributed,p=0.7,list=FALSE)
train_data <- train_sel[c(index),]
test_data <- data.frame(train_sel)[-index,]

## Train models with up sampling method for data balance
# Random Forest model:
ctrl <- trainControl(method = "repeatedcv",
                    number = 5,
                    repeats = 5,
                    verboseIter = FALSE,
                    sampling = "up")
model_rf <- caret::train(is_attributed ~ .,
                        data = train_data,
                        method = "rf",
                        trControl = ctrl)
final_rf <- data.frame(actual = test_data$is_attributed,
                      predict(model_rf, newdata = test_data))

cm_rf_test <- confusionMatrix(final_rf$predict, test_data$is_attributed)
cm_rf_test

```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      1      2
##           1 29907    60
##           2   24     8
##
##           Accuracy : 0.9972
##           95% CI : (0.9965, 0.9978)
##    No Information Rate : 0.9977
##    P-Value [Acc > NIR] : 0.9743531
##
##           Kappa : 0.1588
##
##    McNemar's Test P-Value : 0.0001341
##
##           Sensitivity : 0.9992
##           Specificity : 0.1176
##           Pos Pred Value : 0.9980
##           Neg Pred Value : 0.2500
##           Prevalence : 0.9977
##           Detection Rate : 0.9969
##    Detection Prevalence : 0.9989
##           Balanced Accuracy : 0.5584
##
##           'Positive' Class : 1
##
```

```
# Knn model:
model_knn <- caret::train(is_attributed ~ .,
                           data = train_data,
                           method = "knn",
                           trControl = ctrl)
final_knn <- data.frame(actual = test_data$is_attributed,
                        predict(model_knn, newdata = test_data))

cm_knn_test <- confusionMatrix(final_knn$predict, test_data$is_attributed)
cm_knn_test
```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction      1      2
##           1 29751     51
##           2   180     17
##
##           Accuracy : 0.9923
##           95% CI : (0.9912, 0.9933)
##           No Information Rate : 0.9977
##           P-Value [Acc > NIR] : 1
##
##           Kappa : 0.1254
##
##           McNemar's Test P-Value : <2e-16
##
##           Sensitivity : 0.99399
##           Specificity : 0.25000
##           Pos Pred Value : 0.99829
##           Neg Pred Value : 0.08629
##           Prevalence : 0.99773
##           Detection Rate : 0.99173
##           Detection Prevalence : 0.99343
##           Balanced Accuracy : 0.62199
##
##           'Positive' Class : 1
##

```