**DeteccaoFraudesCliques.R**

Rafael

2019-04-25

*# Projeto Final 1 - Detecção de Fraudes em Cliques de Propaganda Mobile*  
  
*# Legenda Variáveis Dataset*  
  
*#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.*  
  
  
*################ ETAPA 1: CARREGANDO O DATASET E IMPORTANDO BIBLIOTECAS NECESSÁRIAS ################*  
  
dataset <- **read.csv**(file = "{File\_directory}",  
 sep = ",")  
  
**str**(dataset)

## '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 : Factor w/ 80350 levels "2017-11-06 16:00:00",..: 17416 23124 27845 11509 70546 59756 59657 18083 43782 48178 ...  
## $ attributed\_time: Factor w/ 228 levels "","2017-11-06 17:19:04",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ is\_attributed : int 0 0 0 0 0 0 0 0 0 0 ...

*#install.packages("dplyr")*  
*#install.packages("ggplot2")*  
*#install.packages("DMwR")*  
*#install.packages("randomForest")*  
*#install.packages("caret")*  
*#install.packages("caTools")*  
*#install.packages("e1071")*  
*#install.packages("ROCR")*  
**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**(ggplot2)  
**library**(DMwR)

## Loading required package: lattice

## Loading required package: grid

**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

**library**(caret)  
**library**(caTools)  
**library**(e1071)  
**library**(ROCR)

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

*# Verificando se existem dados missing*  
**any**(**is.na**(dataset))

## [1] FALSE

*################ ETAPA 2: GERANDO PLOTS PARA ANÁLISE EXPLORÁTÓRIA ################*  
  
*# Contagem de dados*  
dataset **%>%**  
 **count**(is\_attributed)

## # A tibble: 2 x 2  
## is\_attributed n  
## <int> <int>  
## 1 0 99773  
## 2 1 227

**prop.table**(**table**(dataset**$**is\_attributed))

##   
## 0 1   
## 0.99773 0.00227

*# Realizando uma análise exploratória para identificar quantidade de registros que resultaram em Download ou não.*  
**ggplot**(dataset, **aes**(x = dataset**$**is\_attributed)) **+**   
 **geom\_bar**(**aes**(fill = dataset**$**is\_attributed)) **+**  
 **ggtitle**("Clicks Downloaded / Not Downloaded") **+**  
 **xlab**("Download realizado") **+** **ylab**("Quantidade")



dataset **%>%** **filter**(is\_attributed **==** 1) **%>%**  
**ggplot**(**aes**(x = **as.Date**(click\_time))) **+**   
 **geom\_bar**() **+**  
 **ggtitle**("Quantidade de Downloads por Data")



**head**(dataset)

## 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

*################ ETAPA 3: FEATURE ENGINEERING: TRANSFORMANDO DADOS CATEGÓRICOS E NORMALIZAÇÃO DE DADOS NUMÉRICOS ################*  
  
*# Transformando colunas categóricas*  
var\_categoricas <- **c**("is\_attributed") *#c("ip", "app", "device", "os", "channel", "is\_attributed")*  
  
func\_to.factors <- **function**(dataset, var\_categoricas){  
 **for** (n **in** var\_categoricas){  
 dataset[[n]] <- **as.factor**(dataset[[n]])  
 }  
 **return**(dataset)  
}  
  
dataset\_formated <- **func\_to.factors**(dataset, var\_categoricas)  
  
*# Normalizando os dados numéricos*  
vars\_to\_scale <- **c**("ip", "app", "device", "os", "channel")  
  
func\_scale.features <- **function**(dataset, vars\_to\_scale){  
 **for** (n **in** vars\_to\_scale){  
 dataset[[n]] <- **scale**(dataset[[n]], center=T, scale=T) *# Função scale está sendo utilizada para realizar a normalização dos dados*  
 }  
 **return**(dataset)  
}  
  
dataset\_formated <- **func\_scale.features**(dataset\_formated, vars\_to\_scale)  
  
**str**(dataset\_formated)

## 'data.frame': 100000 obs. of 8 variables:  
## $ ip : num [1:100000, 1] -0.0532 0.2048 0.1456 0.0477 -0.3271 ...  
## ..- attr(\*, "scaled:center")= num 91256  
## ..- attr(\*, "scaled:scale")= num 69836  
## $ app : num [1:100000, 1] -0.0032 0.8669 -0.0032 0.0637 -0.0032 ...  
## ..- attr(\*, "scaled:center")= num 12  
## ..- attr(\*, "scaled:scale")= num 14.9  
## $ device : num [1:100000, 1] -0.08 -0.08 -0.08 -0.08 -0.08 ...  
## ..- attr(\*, "scaled:center")= num 21.8  
## ..- attr(\*, "scaled:scale")= num 260  
## $ os : num [1:100000, 1] -0.1755 -0.104 -0.0683 -0.1755 -0.39 ...  
## ..- attr(\*, "scaled:center")= num 22.8  
## ..- attr(\*, "scaled:scale")= num 55.9  
## $ channel : num [1:100000, 1] 1.7589 -0.0758 -0.4381 1.6047 -0.7002 ...  
## ..- attr(\*, "scaled:center")= num 269  
## ..- attr(\*, "scaled:scale")= num 130  
## $ click\_time : Factor w/ 80350 levels "2017-11-06 16:00:00",..: 17416 23124 27845 11509 70546 59756 59657 18083 43782 48178 ...  
## $ attributed\_time: Factor w/ 228 levels "","2017-11-06 17:19:04",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ is\_attributed : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

*# Formatando as colunas de data*  
dataset\_formated**$**click\_time <- **as.factor**(**as.Date**(dataset\_formated**$**click\_time))  
dataset\_formated**$**attributed\_time <- NULL  
  
final\_dataset <- dataset\_formated  
  
*################ ETAPA 4: ANALISANDO E APLICANDO FEATURE SELECTION ################*   
  
*# Utilizando o algoritmo randomForest para Obter as variaveis mais relevantes do dataset*  
importance\_vars <- **randomForest**(is\_attributed **~** .,   
 data = final\_dataset,   
 ntree = 100,   
 nodesize = 10,   
 importance = TRUE)  
  
**varImp**(importance\_vars)

## 0 1  
## ip 8.6579953 8.6579953  
## app 20.8096392 20.8096392  
## device 5.3371647 5.3371647  
## os 9.0616964 9.0616964  
## channel 14.9571209 14.9571209  
## click\_time 0.6528356 0.6528356

importance\_vars

##   
## Call:  
## randomForest(formula = is\_attributed ~ ., data = final\_dataset, ntree = 100, nodesize = 10, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 100  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 0.2%  
## Confusion matrix:  
## 0 1 class.error  
## 0 99757 16 0.000160364  
## 1 184 43 0.810572687

*################ ETAPA 5: SPLIT DOS DADOS EM TREINO E TESTE E CRIAÇÃO DO MODELO UTILIZANDO O ALGORITMO LINEAR LOGISTIC ################*   
  
*# Split dados em Treino e Teste*  
trainindex <- **sample.split**(final\_dataset**$**is\_attributed, SplitRatio = 0.65)  
dados\_treino <- **subset**(final\_dataset, trainindex **==** TRUE)  
dados\_teste <- **subset**(final\_dataset, trainindex **==** FALSE)  
  
*################ ETAPA 6: CRIANDO MODELOS ################*   
  
*# Modelo 1: Criando um modelo utilizando regressão Logistica*  
  
*# O resultado do modelo de Regressão Logística consiste em "probabilidade" de um evento acontecer devido ao "response".*  
*# O resultado varia entre 0-1 e iremos realizar um arredondamento nos valores para os valores serem 0 ou 1,*  
*# para posteriormente ser possível realizar a comparação do resultado através do método confusionMatrix.*  
linear\_logistic\_model <- **glm**(is\_attributed **~** .,   
 data = dados\_treino,   
 family = **binomial**(link = "logit"))  
  
linear\_logistic\_predict <- **predict**(linear\_logistic\_model,   
 newdata = dados\_teste,   
 type = "response")  
  
linear\_logistic\_predict <- **round**(linear\_logistic\_predict)  
  
**confusionMatrix**(**table**(data = linear\_logistic\_predict,   
 reference = dados\_teste[,7]),   
 positive = '1')

## Confusion Matrix and Statistics  
##   
## reference  
## data 0 1  
## 0 34917 78  
## 1 4 1  
##   
## Accuracy : 0.9977   
## 95% CI : (0.9971, 0.9981)  
## No Information Rate : 0.9977   
## P-Value [Acc > NIR] : 0.6591   
##   
## Kappa : 0.0235   
##   
## Mcnemar's Test P-Value : 7.536e-16   
##   
## Sensitivity : 1.266e-02   
## Specificity : 9.999e-01   
## Pos Pred Value : 2.000e-01   
## Neg Pred Value : 9.978e-01   
## Prevalence : 2.257e-03   
## Detection Rate : 2.857e-05   
## Detection Prevalence : 1.429e-04   
## Balanced Accuracy : 5.063e-01   
##   
## 'Positive' Class : 1   
##

*# Resultado accuracy Modelo 1: 0,9977 Modelo acertou quase todas as classificações resultantes em 0 porém errou todas com 1.*  
  
*# Modelo 2: Criando um modelo utilizando Random Forest*  
  
random\_forest\_model <- **randomForest**(is\_attributed **~** .,   
 data = dados\_treino,   
 method = "class")  
  
radomForest\_predict <- **predict**(random\_forest\_model,   
 newdata = dados\_teste,   
 type = "class")  
  
**confusionMatrix**(**table**(data = radomForest\_predict,   
 reference = dados\_teste[,7]),   
 positive = '1')

## Confusion Matrix and Statistics  
##   
## reference  
## data 0 1  
## 0 34912 65  
## 1 9 14  
##   
## Accuracy : 0.9979   
## 95% CI : (0.9973, 0.9983)  
## No Information Rate : 0.9977   
## P-Value [Acc > NIR] : 0.311   
##   
## Kappa : 0.2738   
##   
## Mcnemar's Test P-Value : 1.62e-10   
##   
## Sensitivity : 0.1772152   
## Specificity : 0.9997423   
## Pos Pred Value : 0.6086957   
## Neg Pred Value : 0.9981416   
## Prevalence : 0.0022571   
## Detection Rate : 0.0004000   
## Detection Prevalence : 0.0006571   
## Balanced Accuracy : 0.5884787   
##   
## 'Positive' Class : 1   
##

*# Resultado accuracy Modelo 2: 0,998 Modelo melhorou a precisão para resultados 1.*  
  
*# É possível analisar através da confusionMatrix, que os algoritmos conseguiram resolver muito bem os registros que onde a classificação final é igual 0.*  
*# Essa informação pode ser visto em "Sensitivity" e "Specificity".*  
*# Isso ocorreu por causa do volume de dados no dataset ser tendêncioso em relação a classificação 0.*  
  
*# Para tentar solucionar esse problema, irei aplicar a ingestão de dados utizando metodos SMOTE e ROSE.*  
  
*################ ETAPA 4: REALIZANDO O BALANCEAMENTO UTILIZANDO SMOTE ################*   
  
*# Balanceando os dados*   
final\_dataset\_Smotted <- **SMOTE**(is\_attributed **~** . , final\_dataset, perc.over = 20000,perc.under=200)  
  
**prop.table**(**table**(final\_dataset\_Smotted**$**is\_attributed))

##   
## 0 1   
## 0.6655574 0.3344426

**table**(final\_dataset\_Smotted**$**is\_attributed)

##   
## 0 1   
## 90800 45627

*################ ETAPA 5: APLICAR O SPLIT DOS DADOS EM TREINO E TESTE NOVAMENTE ################*   
  
*# Split dados em Treino e Teste*  
trainindex\_Smotted <- **sample.split**(final\_dataset\_Smotted**$**is\_attributed, SplitRatio = 0.65)  
dados\_treino\_Smotted <- **subset**(final\_dataset\_Smotted, trainindex **==** TRUE)  
dados\_teste\_Smotted <- **subset**(final\_dataset\_Smotted, trainindex **==** FALSE)  
  
  
linear\_logistic\_model\_smotted <- **glm**(is\_attributed **~** .,   
 data = dados\_treino\_Smotted,   
 family = **binomial**(link = "logit"))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

linear\_logistic\_predict\_smotted <- **predict**(linear\_logistic\_model\_smotted,   
 newdata = dados\_teste\_Smotted,   
 type = "response")  
  
linear\_logistic\_predict\_smotted <- **round**(linear\_logistic\_predict\_smotted)  
  
**confusionMatrix**(**table**(data = linear\_logistic\_predict\_smotted,   
 reference = dados\_teste\_Smotted[,7]),   
 positive = '1')

## Confusion Matrix and Statistics  
##   
## reference  
## data 0 1  
## 0 29215 6064  
## 1 2602 9824  
##   
## Accuracy : 0.8183   
## 95% CI : (0.8149, 0.8218)  
## No Information Rate : 0.667   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5675   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.6183   
## Specificity : 0.9182   
## Pos Pred Value : 0.7906   
## Neg Pred Value : 0.8281   
## Prevalence : 0.3330   
## Detection Rate : 0.2059   
## Detection Prevalence : 0.2605   
## Balanced Accuracy : 0.7683   
##   
## 'Positive' Class : 1   
##

*# Resultado accuracy Modelo 1: 0,8169 A accuracy abaixou, porém melhorou a taxa de acerto resultantes em 0 e 1.*  
  
*# Modelo 2: Criando um modelo utilizando Random Forest*  
  
random\_forest\_model\_smotted <- **randomForest**(is\_attributed **~** .,   
 data = dados\_treino\_Smotted,   
 method = "class")  
  
radomForest\_predict\_smotted <- **predict**(random\_forest\_model\_smotted,   
 newdata = dados\_teste\_Smotted,   
 type = "class")  
  
**confusionMatrix**(**table**(data = radomForest\_predict\_smotted,   
 reference = dados\_teste\_Smotted[,7]),   
 positive = '1')

## Confusion Matrix and Statistics  
##   
## reference  
## data 0 1  
## 0 31726 54  
## 1 91 15834  
##   
## Accuracy : 0.997   
## 95% CI : (0.9964, 0.9974)  
## No Information Rate : 0.667   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9932   
##   
## Mcnemar's Test P-Value : 0.002793   
##   
## Sensitivity : 0.9966   
## Specificity : 0.9971   
## Pos Pred Value : 0.9943   
## Neg Pred Value : 0.9983   
## Prevalence : 0.3330   
## Detection Rate : 0.3319   
## Detection Prevalence : 0.3338   
## Balanced Accuracy : 0.9969   
##   
## 'Positive' Class : 1   
##

*# Resultado accuracy Modelo 2: 0,9965 com uma excelente acertividade para ambas classificações.*   
  
  
*################ ETAPA 6: GERANDO UM GRÁFICO DE CURVA ROC ################*   
  
*# Gerando previsões nos dados de teste*  
*# Criando um dataframe com os dados ORIGINAIS do TESTE junto com a previsão do modelo utilizando os dados de teste.*  
*# com esse dataset poderemos criar uma confusion matrix para analisar a taxa de acerto do modelo de classificação randomForest.*  
df\_previsoes\_smotted <- **data.frame**(observado = dados\_teste\_Smotted**$**is\_attributed,  
 previsto = radomForest\_predict\_smotted)  
  
  
pred <- **prediction**(**as.numeric**(df\_previsoes\_smotted**$**previsto), **as.numeric**(df\_previsoes\_smotted**$**observado))  
perf <- **performance**(pred, "tpr","fpr")   
**plot**(perf, col = **rainbow**(10))

