Projeto01.R

rodrigolima82

2019-05-19

```r
# Script para checar as colunas do dataset

# Carregando os Pacotes
library(data.table)

# Carregando o Dataset #
# Os dados brutos contém 100.000 linhas e 8 colunas (atributos).
# A coluna "is_attributed" é o alvo.
dt <- fread("dados/train_sample.csv")
df <- as.data.frame(dt)

# Remove dt
rm('dt')

# Visualizando dados do dataframe
# View(df)
str(df)
```

```
## '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
# Nome das variáveis
# ip, app, device, os, channel, click_time, attributed_time, is_attributed

# Aplicando Engenharia de Atributos em Variaveis Numericas

# Transformar o objeto de data
df$click_time <- as.POSIXct(df$click_time)

# Extraindo dia e hora
df$click_day <- as.integer(format(df$click_time, "%d"))
df$click_hour <- as.integer(format(df$click_time, "%H"))

# Transformando variáveis numéricas em variáveis categóricas
df$is_attributed <- as.factor(df$is_attributed)


# Remover colunas nao utilizadas
df$click_time <- NULL
df$attributed_time <- NULL
```

```r
str(df)
```

```
## '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 ...
##  $ is_attributed: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ click_day    : int  7 7 7 7 9 9 9 7 8 8 ...
##  $ click_hour   : int  9 13 18 4 9 1 1 10 9 12 ...
```

```r
# Analise Exploratoria de Dados

# Carregando os Pacotes
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```r
# Verificar se existem valores ausentes (missing) em cada coluna
# Nenhum valor encontrado
any(is.na(df))
```

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

```r
# Analisando dados por agrupamentos
df %>% group_by(ip) %>% tally()
```

```
## # A tibble: 34,857 x 2
##       ip     n
##    <int> <int>
##  1     9     1
##  2    10     3
##  3    19     1
##  4    20     4
##  5    25     1
##  6    27     5
##  7    31     1
##  8    33     1
##  9    36     3
## 10    59     3
## # ... with 34,847 more rows
```

```r
df %>% group_by(ip, click_day) %>% tally()
```

```
## # A tibble: 55,454 x 3
## # Groups:   ip [34,857]
##       ip click_day     n
##    <int>     <int> <int>
##  1     9         7     1
##  2    10         7     2
##  3    10         8     1
##  4    19         8     1
##  5    20         8     3
##  6    20         9     1
##  7    25         7     1
##  8    27         7     1
##  9    27         8     2
## 10    27         9     2
## # ... with 55,444 more rows
```

```r
# Adicionando nova coluna no dataframe
df_count_ip <- df %>%
  count(ip, sort = TRUE, name = "ip_count")

df <- merge(df, df_count_ip, by=c("ip"))
rm('df_count_ip')

df_count_ip_day <- df %>%
  count(ip, click_day, sort = TRUE, name = "ip_day_count")

df <- merge(df, df_count_ip_day, by=c('ip','click_day'))
rm('df_count_ip_day')

str(df)
```

```
## 'data.frame':    100000 obs. of  10 variables:
##  $ ip           : int  10 10 10 1000 100002 100005 100005 100009 100013 100013 ...
##  $ click_day    : int  7 7 8 7 8 7 9 8 8 9 ...
##  $ app          : int  11 12 18 12 3 2 9 64 3 13 ...
##  $ device       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ os           : int  22 19 13 19 41 17 19 18 41 10 ...
##  $ channel      : int  319 140 107 178 280 219 232 459 442 477 ...
##  $ is_attributed: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ click_hour   : int  1 7 11 13 2 3 14 9 5 7 ...
##  $ ip_count     : int  3 3 3 1 1 2 2 1 2 2 ...
##  $ ip_day_count : int  2 2 1 1 1 1 1 1 1 1 ...
```

```r
# View(df)

# Normalizar as variaveis numericas
cols <- c('ip', 'app', 'device', 'os', 'channel','click_day','click_hour','ip_count', 'ip_day_count')
df[, cols] <- scale(df[, cols])

# Verificando overfitting dos dados
# 99.773 registros indicam que o app nao foi baixado
# 227 registros indicam que o app foi baixado
table(df$is_attributed)
```

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

```
## 99773   227
```

```r
prop.table(table(df$is_attributed))
```

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

```r
# Feature Selection (Selecao de Variaveis)

# Carregando os Pacotes
library(ROSE)
```

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

```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```r
library(e1071)
library(rpart)

# Gerando dados de treino e de teste
splits <- createDataPartition(df$is_attributed, p=0.7, list=FALSE)

# Separando os dados
dados_treino <- df[ splits,]
dados_teste <- df[-splits,]

# Verificando o numero de linhas
nrow(dados_treino)
```

```
## [1] 70001
```

```r
nrow(dados_teste)
```

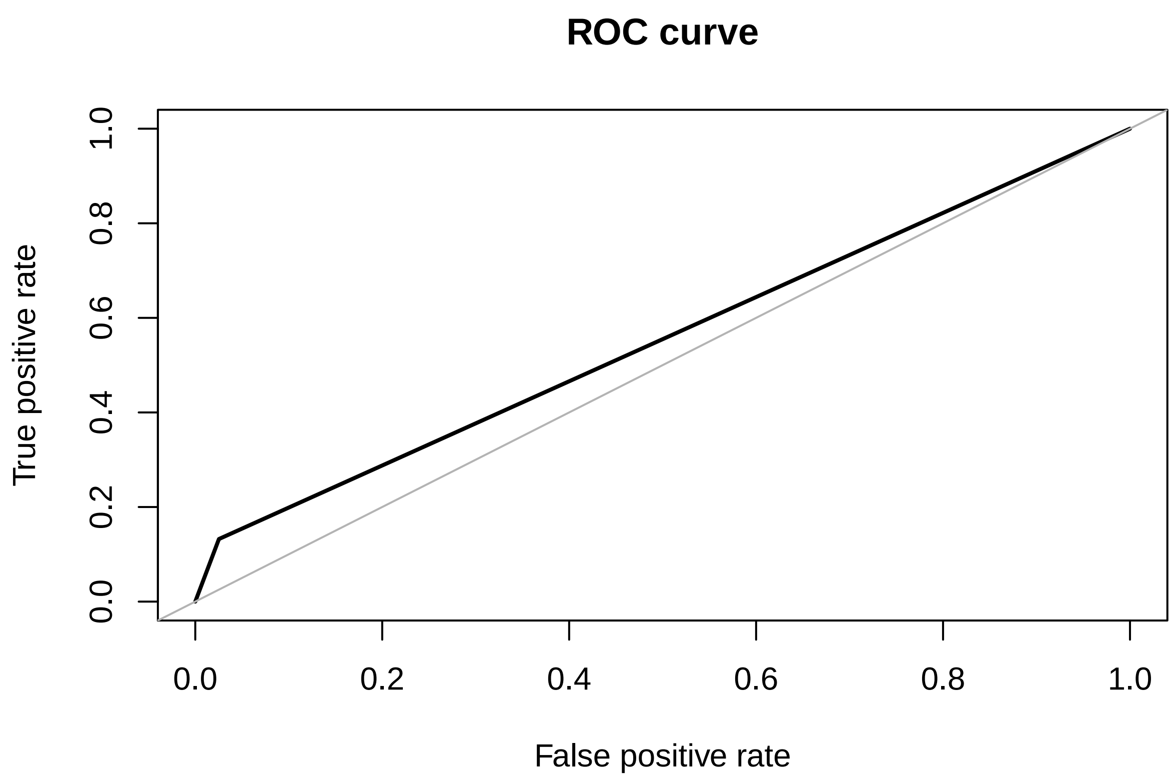
```
## [1] 29999
```

```r
# Treinando o modelo usando Naive Bayes e fazendo predicoes
## devido ao problema de overfitting, o resultado esta tendencioso
## necessario corrigir o problema de overfitting
modeloNB <- naiveBayes(is_attributed ~. , data=dados_treino)
predNB <- predict(modeloNB, dados_teste)
confusionMatrix(predNB, dados_teste$is_attributed)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 29173    59
##          1   758     9
```

```
##
##                Accuracy : 0.9728
##                  95% CI : (0.9709, 0.9746)
##     No Information Rate : 0.9977
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0175
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.97468
##             Specificity : 0.13235
##          Pos Pred Value : 0.99798
##          Neg Pred Value : 0.01173
##              Prevalence : 0.99773
##          Detection Rate : 0.97247
##    Detection Prevalence : 0.97443
##       Balanced Accuracy : 0.55351
##
##        'Positive' Class : 0
##
```

```
# AUC
roc.curve(dados_teste$is_attributed, predNB)
```




```
## Area under the curve (AUC): 0.554
```

```
# Resolvendo problema de Overfitting usando pacote ROSE
#over sampling
dados_treino_new <- ROSE(is_attributed ~ . , data=dados_treino)$data
table(dados_treino_new$is_attributed)
```

```
##
##     0     1
## 35177 34824
```

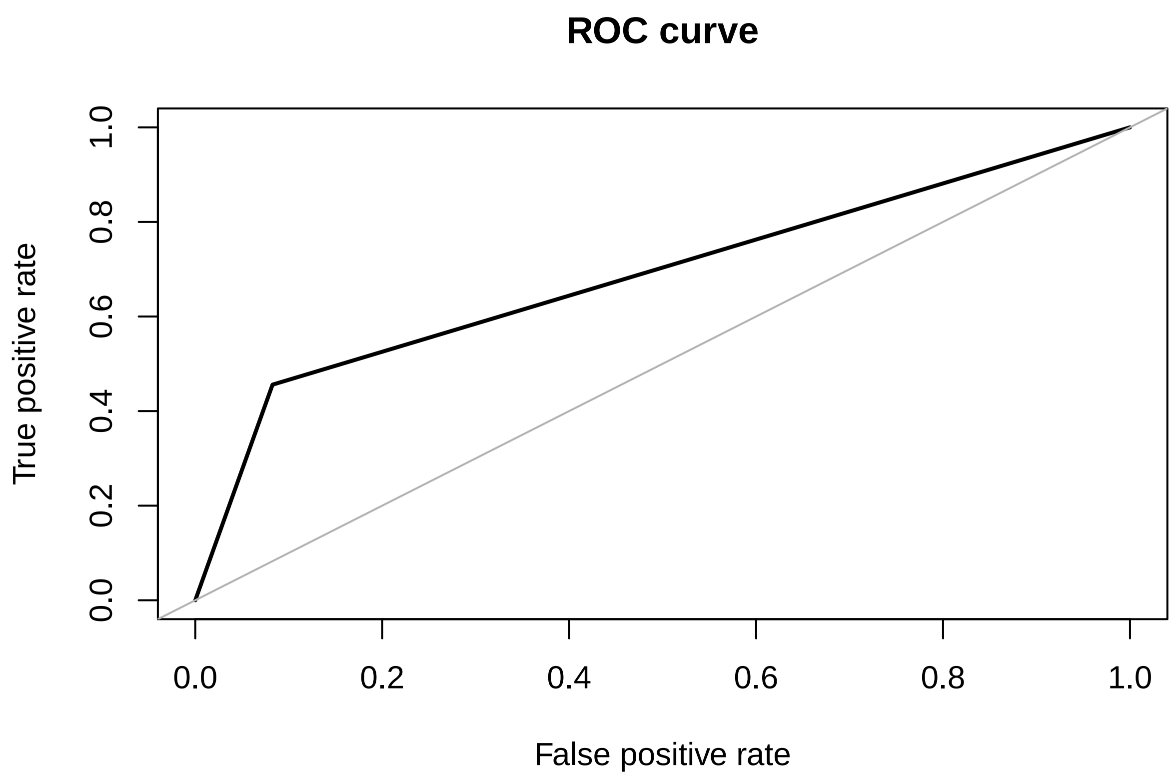
```r
prop.table(table(dados_treino_new$is_attributed))
```

```
##
##         0         1
## 0.5025214 0.4974786
```

```r
# Treinando um novo modelo com os novos dados de treino
modeloNB_v2 <- naiveBayes(is_attributed ~ . , data=dados_treino_new)
predNB_v2 <- predict(modeloNB_v2, dados_teste)
confusionMatrix(predNB_v2, dados_teste$is_attributed)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 27456    37
##          1  2475    31
##
##                Accuracy : 0.9163
##                  95% CI : (0.9131, 0.9194)
##     No Information Rate : 0.9977
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0198
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.91731
##             Specificity : 0.45588
##          Pos Pred Value : 0.99865
##          Neg Pred Value : 0.01237
##              Prevalence : 0.99773
##          Detection Rate : 0.91523
##    Detection Prevalence : 0.91646
##       Balanced Accuracy : 0.68660
##
##        'Positive' Class : 0
##
```

```r
# AUC
roc.curve(dados_teste$is_attributed, predNB_v2)
```




```
## Area under the curve (AUC): 0.687
```

```r
#AUC ROSE
ROSE.holdout <- ROSE.eval(is_attributed ~ .,
                          data = dados_treino_new,
                          learner = rpart,
                          method.assess = "holdout",
                          extr.pred = function(obj)obj[,2])
ROSE.holdout
```

```
##
## Call:
## ROSE.eval(formula = is_attributed ~ ., data = dados_treino_new,
##     learner = rpart, extr.pred = function(obj) obj[, 2], method.assess = "holdout")
##
## Holdout estimate of auc: 0.846
```

```r
# Análise de Correlação

# Carregando os Pacotes
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(corrgram)
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```
##
## Attaching package: 'corrgram'
```

```
## The following object is masked from 'package:lattice':
##
##     panel.fill
```

```r
# obtendo somente as colunas numericas
colunas_numericas <- sapply(dados_treino_new, is.numeric)
colunas_numericas
```

```
##            ip     click_day           app        device            os
##          TRUE          TRUE          TRUE          TRUE          TRUE
##       channel is_attributed    click_hour      ip_count  ip_day_count
##          TRUE         FALSE          TRUE          TRUE          TRUE
```

```r
# Filtrando as colunas numericas para correlacao
data_cor <- cor(dados_treino_new[,colunas_numericas])
data_cor
```

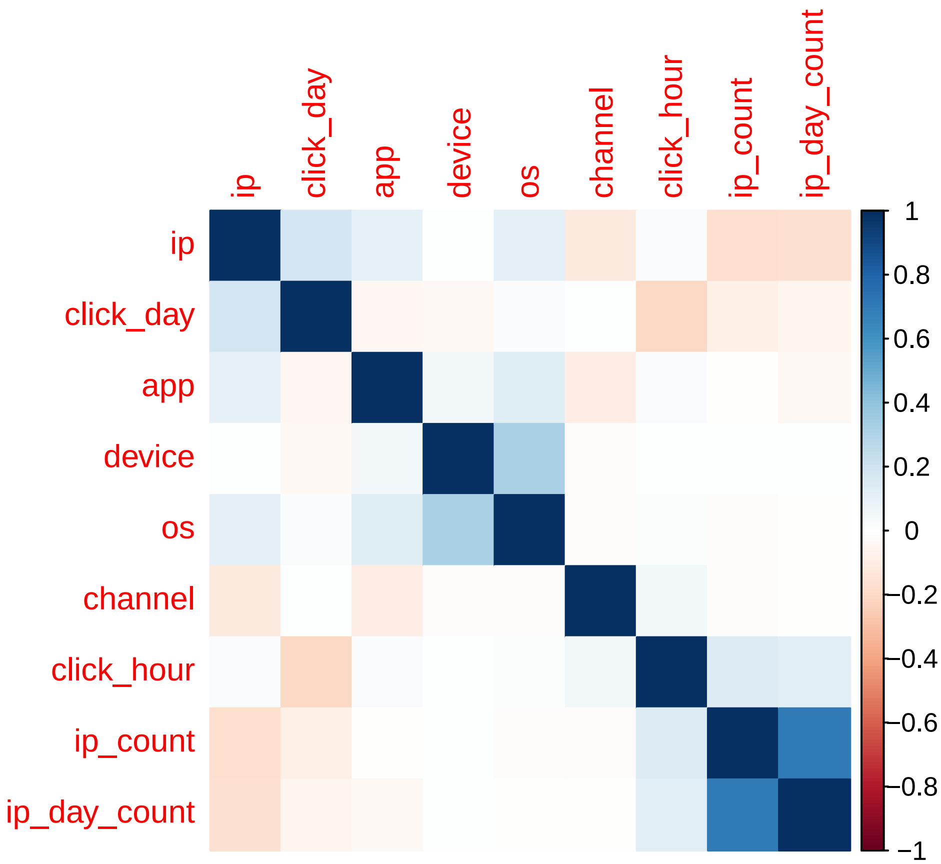
```
##                        ip    click_day          app       device
## ip            1.000000000  0.188812587  0.105519985  0.004707135
## click_day     0.188812587  1.000000000 -0.046530254 -0.031124922
## app           0.105519985 -0.046530254  1.000000000  0.053060671
## device        0.004707135 -0.031124922  0.053060671  1.000000000
## os            0.119795925  0.028070439  0.132279079  0.329260697
## channel      -0.116477683  0.004050537 -0.093331558 -0.015940436
## click_hour    0.026053281 -0.208323592  0.026232579  0.004416267
## ip_count     -0.176095858 -0.081155777 -0.003287766  0.002044292
## ip_day_count -0.161268845 -0.052111784 -0.030515931  0.004682263
##                        os       channel   click_hour     ip_count
## ip            0.119795925 -0.1164776830  0.026053281 -0.176095858
## click_day     0.028070439  0.0040505375 -0.208323592 -0.081155777
## app           0.132279079 -0.0933315584  0.026232579 -0.003287766
## device        0.329260697 -0.0159404360  0.004416267  0.002044292
## os            1.000000000 -0.0172733148  0.014947085 -0.012736725
## channel      -0.017273315  1.0000000000  0.056284018 -0.013373677
## click_hour    0.014947085  0.0562840184  1.000000000  0.145328667
## ip_count     -0.012736725 -0.0133736772  0.145328667  1.000000000
## ip_day_count -0.008899855 -0.0002934833  0.123115635  0.704119166
##               ip_day_count
## ip           -0.1612688446
## click_day    -0.0521117839
## app          -0.0305159312
## device        0.0046822627
## os           -0.0088998554
## channel      -0.0002934833
## click_hour    0.1231156351
## ip_count      0.7041191661
## ip_day_count  1.0000000000
```

```r
head(data_cor)
```

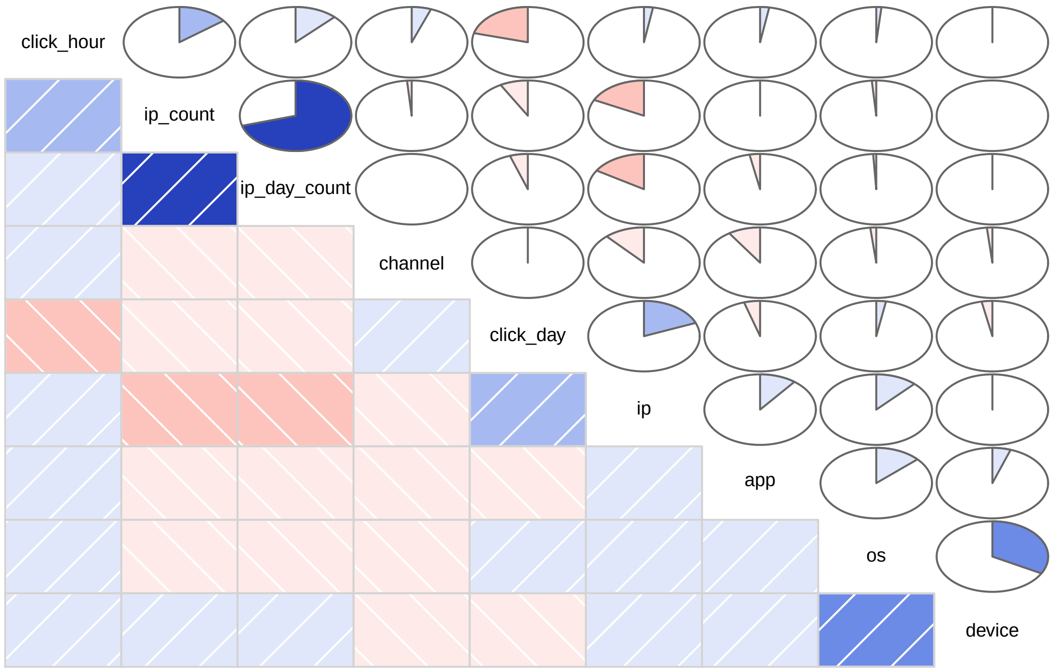
```
##                     ip    click_day         app       device          os
## ip         1.000000000  0.188812587  0.10551999  0.004707135  0.11979592
## click_day  0.188812587  1.000000000 -0.04653025 -0.031124922  0.02807044
## app        0.105519985 -0.046530254  1.00000000  0.053060671  0.13227908
## device     0.004707135 -0.031124922  0.05306067  1.000000000  0.32926070
## os         0.119795925  0.028070439  0.13227908  0.329260697  1.00000000
```

```
## channel   -0.116477683  0.004050537 -0.09333156 -0.015940436 -0.01727331
##                channel   click_hour     ip_count  ip_day_count
## ip        -0.116477683  0.026053281 -0.176095858 -0.1612688446
## click_day  0.004050537 -0.208323592 -0.081155777 -0.0521117839
## app       -0.093331558  0.026232579 -0.003287766 -0.0305159312
## device    -0.015940436  0.004416267  0.002044292  0.0046822627
## os        -0.017273315  0.014947085 -0.012736725 -0.0088998554
## channel    1.000000000  0.056284018 -0.013373677 -0.0002934833
```

```r
# Criando um corrplot
corrplot(data_cor, method = 'color')
```




```r
# Criando um corrgram
corrgram(dados_treino_new, order=TRUE, lower.panel = panel.shade,
         upper.panel = panel.pie, text.panel = panel.txt)
```




```r
# Cria um modelo preditivo usando randomForest

# Carregando os Pacotes
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
# Cria o modelo preditivo usando randomForest
modeloRF <- randomForest(is_attributed ~ .,
                         data = dados_treino_new,
                         ntree = 40,
                         nodesize = 5)
print(modeloRF)
```

```
##
## Call:
##  randomForest(formula = is_attributed ~ ., data = dados_treino_new,      ntree = 40, nodesize = 5)
##                Type of random forest: classification
##                      Number of trees: 40
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 9.45%
## Confusion matrix:
```

```
##       0     1 class.error
## 0 32276  2901  0.08246866
## 1  3715 31109  0.10667930
```

```r
# Previsões com um modelo de classificação baseado em randomForest

# Gerando previsões nos dados de teste
previsoes <- data.frame(observado = dados_teste$is_attributed,
                        previsto = predict(modeloRF, newdata = dados_teste))


# Visualizando o resultado
# View(previsoes)

# Calculando a Confusion Matrix em R

# Carregando os Pacotes
library(ROCR)
```

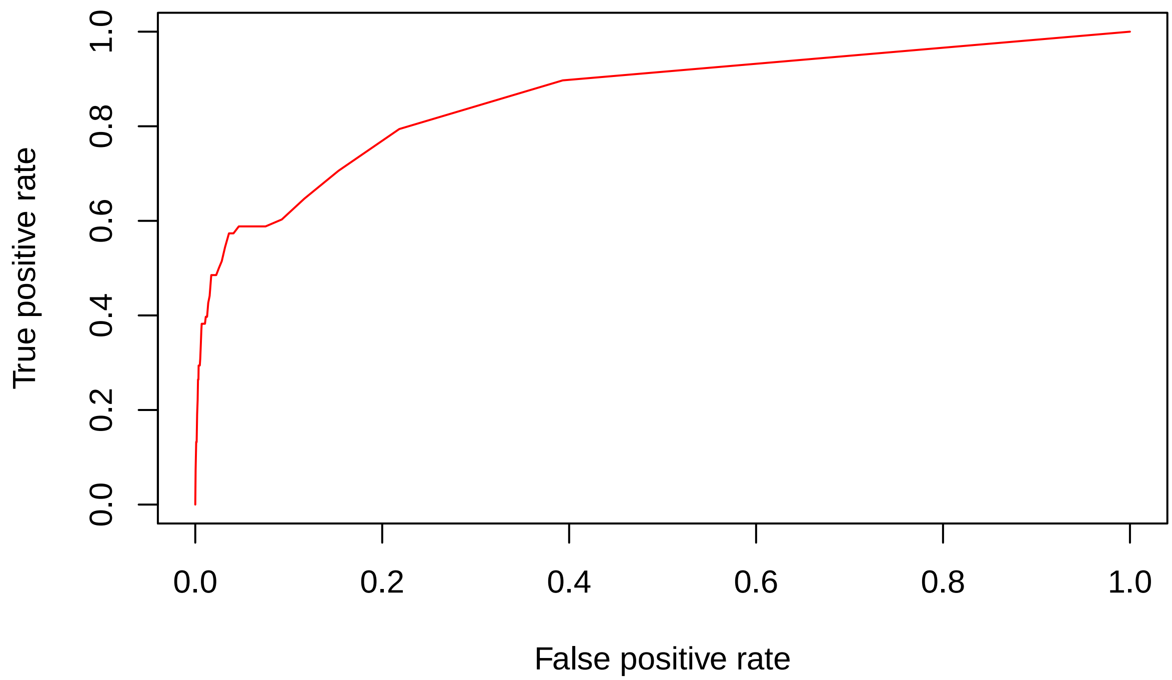
```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

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

```r
# Gerando as classes de dados
class1 <- predict(modeloRF, newdata = dados_teste, type = 'prob')
class2 <- dados_teste$is_attributed

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




```
# Gerando Confusion Matrix com o Caret
# Dataframes com valores observados e previstos
previsoes_v2 <- data.frame(observado = dados_teste$is_attributed,
                           previsto = predict(object = modeloRF, newdata = dados_teste))

confusionMatrix(previsoes_v2$observado, previsoes_v2$previsto)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction      0     1
##          0 29577   354
##          1    41    27
##
##                Accuracy : 0.9868
##                  95% CI : (0.9855, 0.9881)
##     No Information Rate : 0.9873
##     P-Value [Acc > NIR] : 0.7738
##
##                   Kappa : 0.1169
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99862
##             Specificity : 0.07087
##          Pos Pred Value : 0.98817
##          Neg Pred Value : 0.39706
##              Prevalence : 0.98730
##          Detection Rate : 0.98593
##    Detection Prevalence : 0.99773
##       Balanced Accuracy : 0.53474
##
##        'Positive' Class : 0
```

```
##
```

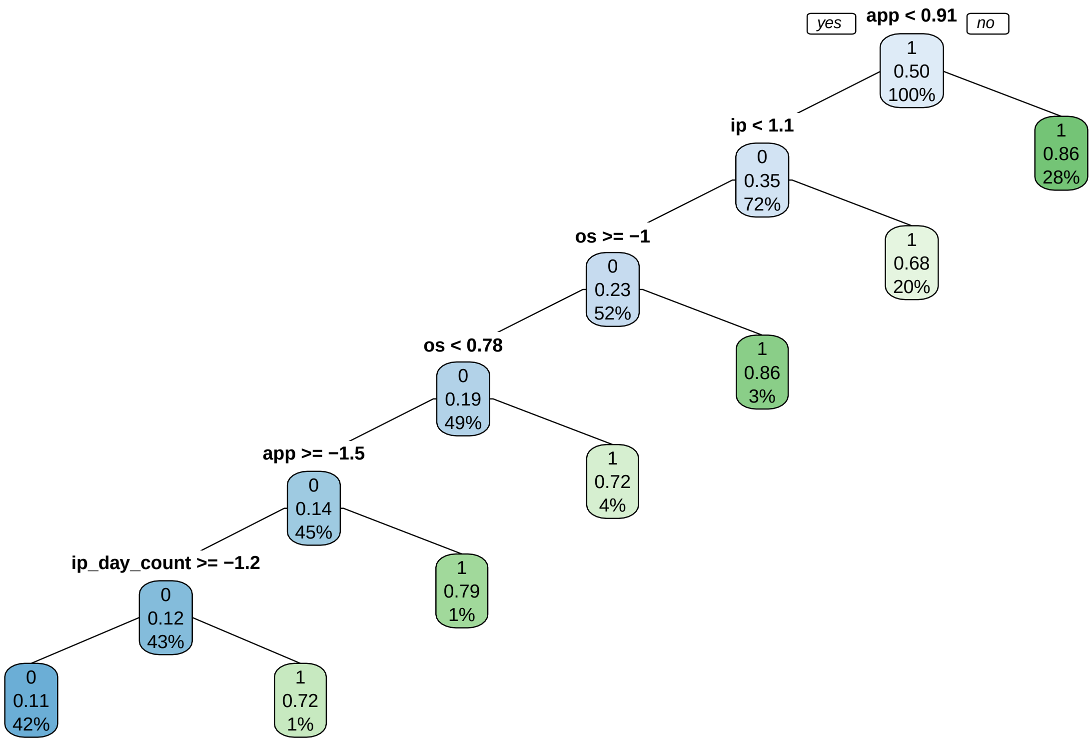
```r
# Otimizando o Modelo preditivo

# Carregando os Pacotes
library(rpart.plot)

# Criando uma Cost Function
Cost_func <- matrix(c(0, 1.5, 1, 0), nrow = 2, dimnames = list(c("1", "2"), c("1", "2")))

# Criando o Modelo usando rpart
modeloTree <- rpart(is_attributed ~ .,
                    data = dados_treino_new,
                    method = 'class',
                    parms = list(loss = Cost_func))

# Plot do modelo
rpart.plot(modeloTree, fallen.leaves = FALSE, type = 1)
```




```r
# Analisando Confusion Matrix
pred.tree <- predict(modeloTree, type = "class")
confusionMatrix(pred.tree, dados_treino_new$is_attributed)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 26467  3209
##          1  8710 31615
##
##                Accuracy : 0.8297
```

```
##                  95% CI : (0.8269, 0.8325)
##     No Information Rate : 0.5025
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6597
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7524
##             Specificity : 0.9079
##          Pos Pred Value : 0.8919
##          Neg Pred Value : 0.7840
##              Prevalence : 0.5025
##          Detection Rate : 0.3781
##    Detection Prevalence : 0.4239
##       Balanced Accuracy : 0.8301
##
##        'Positive' Class : 0
##
```

```
# Modelo usando Naive Bayes com os novos dados de treino
modeloNB_v2 <- naiveBayes(is_attributed ~. , data=dados_treino_new)
predNB_v2 <- predict(modeloNB_v2, dados_teste)
confusionMatrix(predNB_v2, dados_teste$is_attributed)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 27456    37
##          1  2475    31
##
##                Accuracy : 0.9163
##                  95% CI : (0.9131, 0.9194)
##     No Information Rate : 0.9977
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0198
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.91731
##             Specificity : 0.45588
##          Pos Pred Value : 0.99865
##          Neg Pred Value : 0.01237
##              Prevalence : 0.99773
##          Detection Rate : 0.91523
##    Detection Prevalence : 0.91646
##       Balanced Accuracy : 0.68660
##
##        'Positive' Class : 0
##
```

```
# Modelo usando RandomForest com os novos dados de treino
modeloRF_v2 <- randomForest(is_attributed ~ .,
```

```r
                         data = dados_treino_new,
                         ntree = 100,
                         nodesize = 5)
predRF_v2 <- predict(modeloRF_v2, dados_teste)
confusionMatrix(predRF_v2, dados_teste$is_attributed)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 29565    41
##          1   366    27
##
##                Accuracy : 0.9864
##                  95% CI : (0.9851, 0.9877)
##     No Information Rate : 0.9977
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1137
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9878
##             Specificity : 0.3971
##          Pos Pred Value : 0.9986
##          Neg Pred Value : 0.0687
##              Prevalence : 0.9977
##          Detection Rate : 0.9855
##    Detection Prevalence : 0.9869
##       Balanced Accuracy : 0.6924
##
##        'Positive' Class : 0
##
```