Projeto01.R

rodrigolima82

2019-05-19

# 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" "2017-11-07 04:58:08" ...  
## $ attributed\_time: chr "" "" "" "" ...  
## $ is\_attributed : int 0 0 0 0 0 0 0 0 0 0 ...

# 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  
  
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 ...

# 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

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

## [1] FALSE

# 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

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

# 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 ...

# 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

prop.table(table(df$is\_attributed))

##   
## 0 1   
## 0.99773 0.00227

# Feature Selection (Selecao de Variaveis)  
  
# Carregando os Pacotes  
library(ROSE)

## Loaded ROSE 0.0-3

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

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

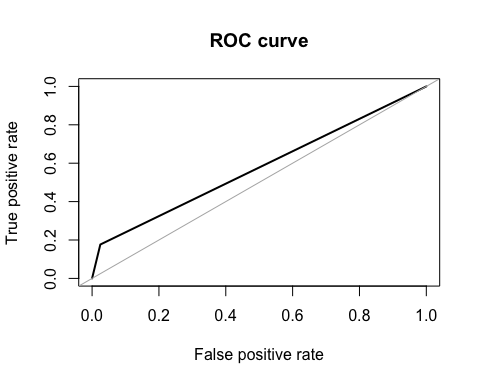
nrow(dados\_teste)

## [1] 29999

# Treinando o modelo usando Baive 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 29193 56  
## 1 738 12  
##   
## Accuracy : 0.9735   
## 95% CI : (0.9717, 0.9753)  
## No Information Rate : 0.9977   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0253   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9753   
## Specificity : 0.1765   
## Pos Pred Value : 0.9981   
## Neg Pred Value : 0.0160   
## Prevalence : 0.9977   
## Detection Rate : 0.9731   
## Detection Prevalence : 0.9750   
## Balanced Accuracy : 0.5759   
##   
## 'Positive' Class : 0   
##

# AUC  
roc.curve(dados\_teste$is\_attributed, predNB)



## Area under the curve (AUC): 0.576

# 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   
## 34659 35342

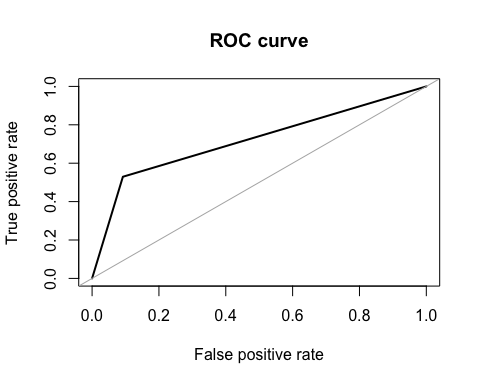
prop.table(table(dados\_treino\_new$is\_attributed))

##   
## 0 1   
## 0.4951215 0.5048785

# 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 27173 32  
## 1 2758 36  
##   
## Accuracy : 0.907   
## 95% CI : (0.9037, 0.9103)  
## No Information Rate : 0.9977   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0208   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.90785   
## Specificity : 0.52941   
## Pos Pred Value : 0.99882   
## Neg Pred Value : 0.01288   
## Prevalence : 0.99773   
## Detection Rate : 0.90580   
## Detection Prevalence : 0.90686   
## Balanced Accuracy : 0.71863   
##   
## 'Positive' Class : 0   
##

# AUC  
roc.curve(dados\_teste$is\_attributed, predNB\_v2)



## Area under the curve (AUC): 0.719

#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: 0.833

# Análise de Correlação   
  
# Carregando os Pacotes  
library(corrplot)

## corrplot 0.84 loaded

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

# 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

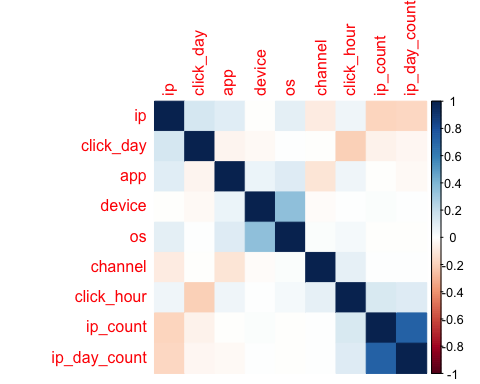
# Filtrando as colunas numericas para correlacao  
data\_cor <- cor(dados\_treino\_new[,colunas\_numericas])  
data\_cor

## ip click\_day app device  
## ip 1.000000000 0.148147746 0.10693157 -0.002627978  
## click\_day 0.148147746 1.000000000 -0.04328420 -0.025494013  
## app 0.106931570 -0.043284204 1.00000000 0.069031762  
## device -0.002627978 -0.025494013 0.06903176 1.000000000  
## os 0.099059332 0.008359963 0.11638869 0.352910157  
## channel -0.085004990 -0.008663318 -0.11865543 -0.010162347  
## click\_hour 0.059458433 -0.208464378 0.05179427 0.003603034  
## ip\_count -0.188855542 -0.059209211 -0.00120148 0.014094193  
## ip\_day\_count -0.175216504 -0.038814143 -0.02712762 0.009923906  
## os channel click\_hour ip\_count  
## ip 0.099059332 -0.085004990 0.059458433 -0.188855542  
## click\_day 0.008359963 -0.008663318 -0.208464378 -0.059209211  
## app 0.116388694 -0.118655431 0.051794271 -0.001201480  
## device 0.352910157 -0.010162347 0.003603034 0.014094193  
## os 1.000000000 0.017588514 0.033113193 -0.006878819  
## channel 0.017588514 1.000000000 0.086705759 0.006877926  
## click\_hour 0.033113193 0.086705759 1.000000000 0.134972440  
## ip\_count -0.006878819 0.006877926 0.134972440 1.000000000  
## ip\_day\_count -0.007938122 0.007991331 0.114482166 0.720266013  
## ip\_day\_count  
## ip -0.175216504  
## click\_day -0.038814143  
## app -0.027127616  
## device 0.009923906  
## os -0.007938122  
## channel 0.007991331  
## click\_hour 0.114482166  
## ip\_count 0.720266013  
## ip\_day\_count 1.000000000

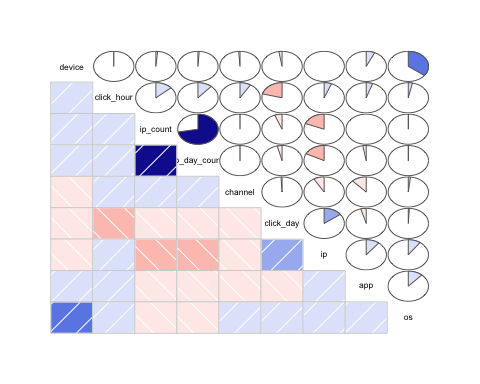
head(data\_cor)

## ip click\_day app device os  
## ip 1.000000000 0.148147746 0.10693157 -0.002627978 0.099059332  
## click\_day 0.148147746 1.000000000 -0.04328420 -0.025494013 0.008359963  
## app 0.106931570 -0.043284204 1.00000000 0.069031762 0.116388694  
## device -0.002627978 -0.025494013 0.06903176 1.000000000 0.352910157  
## os 0.099059332 0.008359963 0.11638869 0.352910157 1.000000000  
## channel -0.085004990 -0.008663318 -0.11865543 -0.010162347 0.017588514  
## channel click\_hour ip\_count ip\_day\_count  
## ip -0.085004990 0.059458433 -0.188855542 -0.175216504  
## click\_day -0.008663318 -0.208464378 -0.059209211 -0.038814143  
## app -0.118655431 0.051794271 -0.001201480 -0.027127616  
## device -0.010162347 0.003603034 0.014094193 0.009923906  
## os 0.017588514 0.033113193 -0.006878819 -0.007938122  
## channel 1.000000000 0.086705759 0.006877926 0.007991331

# Criando um corrplot  
corrplot(data\_cor, method = 'color')



# Criando um corrgram  
corrgram(dados\_treino\_new, order=TRUE, lower.panel = panel.shade,  
 upper.panel = panel.pie, text.panel = panel.txt)



# 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

# 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: 10.72%  
## Confusion matrix:  
## 0 1 class.error  
## 0 31262 3397 0.09801206  
## 1 4107 31235 0.11620735

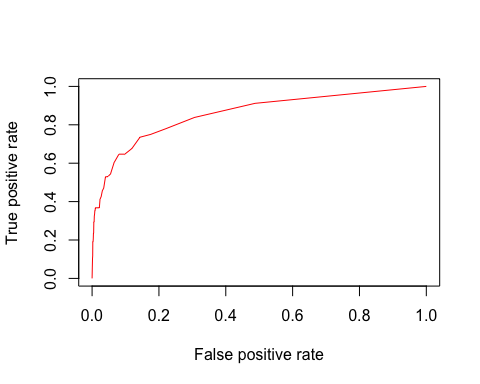
# Previsões com um modelo de classificação baseado em randomForest  
  
# Gerando previsões nos dados de teste  
previsoes <- data.frame(observado = dados\_teste$is\_attributed,  
 previsto = predict(modeloRF, newdata = dados\_teste))  
  
  
# Visualizando o resultado  
# View(previsoes)  
  
# Calculando 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

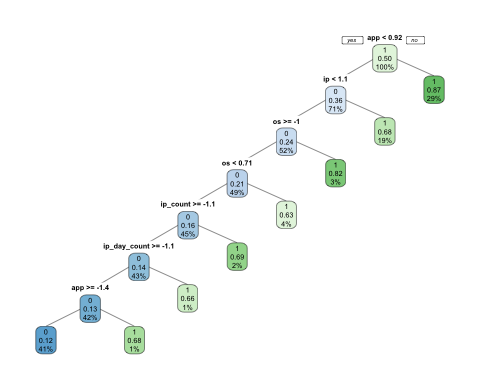
# Gerando as classes de dados  
class1 <- predict(modeloRF, newdata = dados\_teste, type = 'prob')  
class2 <- dados\_teste$is\_attributed  
  
# Gerando a curva ROC  
pred <- prediction(class1[,2], class2)  
perf <- performance(pred, "tpr","fpr")   
plot(perf, col = rainbow(10))



# 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 29452 479  
## 1 43 25  
##   
## Accuracy : 0.9826   
## 95% CI : (0.9811, 0.984)  
## No Information Rate : 0.9832   
## P-Value [Acc > NIR] : 0.7977   
##   
## Kappa : 0.0838   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9985   
## Specificity : 0.0496   
## Pos Pred Value : 0.9840   
## Neg Pred Value : 0.3676   
## Prevalence : 0.9832   
## Detection Rate : 0.9818   
## Detection Prevalence : 0.9977   
## Balanced Accuracy : 0.5241   
##   
## 'Positive' Class : 0   
##

# 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)



# 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 25286 3357  
## 1 9373 31985  
##   
## Accuracy : 0.8181   
## 95% CI : (0.8153, 0.821)  
## No Information Rate : 0.5049   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6356   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7296   
## Specificity : 0.9050   
## Pos Pred Value : 0.8828   
## Neg Pred Value : 0.7734   
## Prevalence : 0.4951   
## Detection Rate : 0.3612   
## Detection Prevalence : 0.4092   
## Balanced Accuracy : 0.8173   
##   
## 'Positive' Class : 0   
##