Projeto 1: Detecção de Fraudes no Tráfego de Cliques em Propagandas de Aplicações Mobile

Camila

06/07/2021

ETAPA 1: Definição do problema

Objetivo: Prever se um usuário fará o download de um app após clicar em um anúncio de um aplicativo móvel.

ETAPA 2: Coleta dos dados

Os dados foram baixados do site kaggle, no endereço abaixo:
https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data

Inicialmente será carregado o arquivo de treino "train_sample", utilizando o pacote 'readr', devido o arquivo ser grande:

ETAPA 3: Análise exploratória

Após os dados serem carregados, os mesmos serão analisados: Em uma visualização prévia, é possível verificar que os dados possuem 8 colunas

```
## # A tibble: 6 x 8
##       ip   app device    os channel click_time
attributed_time
##    <dbl> <dbl>  <dbl> <dbl>   <dbl> <dttm>              <dttm>
## 1  87540    12      1    13     497 2017-11-07 09:30:38 NA
## 2 105560    25      1    17     259 2017-11-07 13:40:27 NA
## 3 101424    12      1    19     212 2017-11-07 18:05:24 NA
## 4  94584    13      1    13     477 2017-11-07 04:58:08 NA
## 5  68413    12      1     1     178 2017-11-09 09:00:09 NA
## 6  93663     3      1    17     115 2017-11-09 01:22:13 NA
## # ... with 1 more variable: is_attributed <dbl>
```

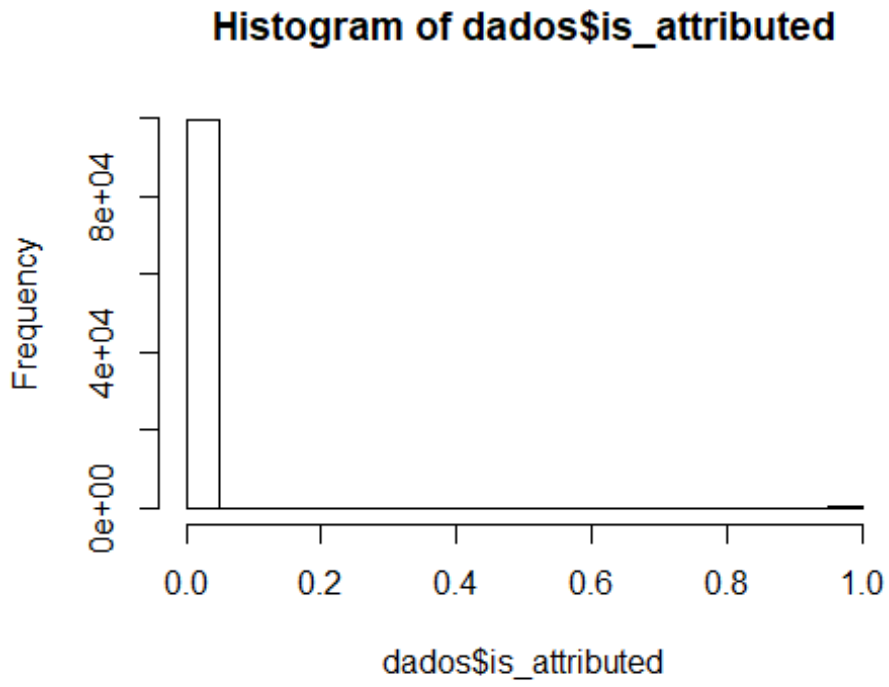
Em uma visualização mais profunda, verifica-se a existência de 100 mil observações (linhas) no conjunto de treino

Verificando a classificação dos dados:

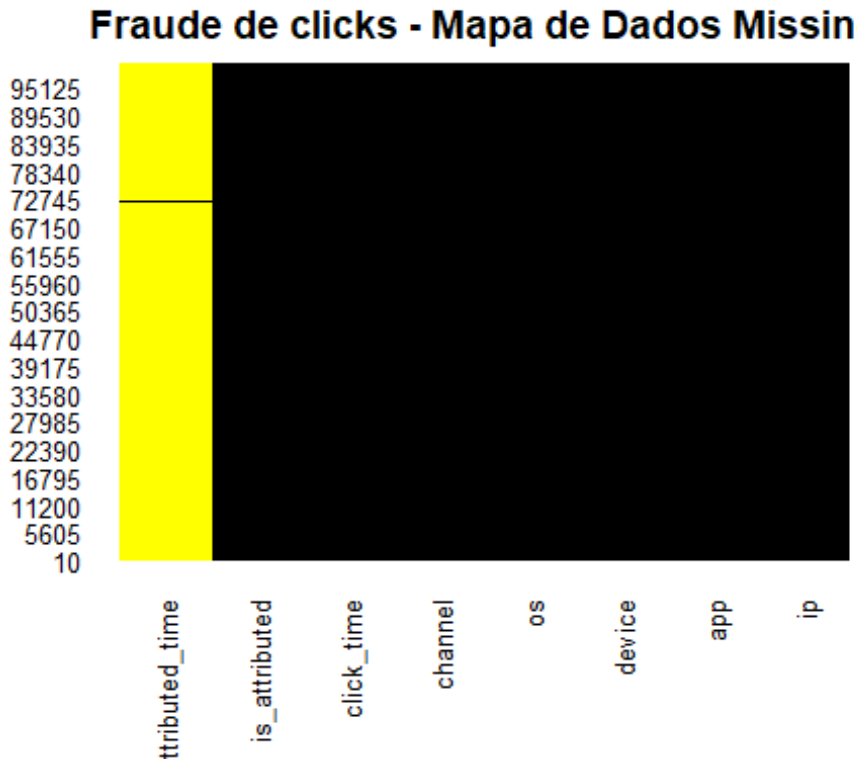
```
## spec_tbl_df[,8] [100,000 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ip             : num [1:100000] 87540 105560 101424 94584 68413 ...
##  $ app            : num [1:100000] 12 25 12 13 12 3 1 9 2 3 ...
##  $ device         : num [1:100000] 1 1 1 1 1 1 1 1 2 1 ...
##  $ os             : num [1:100000] 13 17 19 13 1 17 17 25 22 19 ...
##  $ channel        : num [1:100000] 497 259 212 477 178 115 135 442 364
135 ...
```

```
##  $ click_time     : POSIXct[1:100000], format: "2017-11-07 09:30:38"
"2017-11-07 13:40:27" ...
##  $ attributed_time: POSIXct[1:100000], format: NA NA ...
##  $ is_attributed  : num [1:100000] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. 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()
##   .. )
```

Analisando a distribuição dos dados: Há muito mais dados com zero do que com 1. Sendo assim, será feita uma nova divisão dos dados em treino e teste.



Verificando se existem dados missing no dataset: A única coluna que possui dados missing é a coluna “attributed_time”. Essa coluna está vazia quando o usuário não fez o download. Dessa forma, era provável que isso ocorresse.



Verificando as variáveis mais relevantes no dataset: Para verificar as variáveis importantes, a variável target foi comparada com todas as outras variáveis, exceto a "attributed_time", visto que esta só possui dados, quando "is_attributed" é igual a TRUE.

ETAPA 4: Pré-processamento (se necessário)

Como a última coluna ("is_attributed") é a variável target e está classificada como numérica, a mesma será transformada para o tipo fator:

```
## spec_tbl_df[,8] [100,000 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ip             : num [1:100000] 87540 105560 101424 94584 68413 ...
##  $ app            : num [1:100000] 12 25 12 13 12 3 1 9 2 3 ...
##  $ device         : num [1:100000] 1 1 1 1 1 1 1 1 2 1 ...
##  $ os             : num [1:100000] 13 17 19 13 1 17 17 25 22 19 ...
##  $ channel        : num [1:100000] 497 259 212 477 178 115 135 442 364
135 ...
##  $ click_time     : POSIXct[1:100000], format: "2017-11-07 09:30:38"
"2017-11-07 13:40:27" ...
##  $ attributed_time: POSIXct[1:100000], format: NA NA ...
##  $ is_attributed  : num [1:100000] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. 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()
##   .. )

## spec_tbl_df[,8] [100,000 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ip             : num [1:100000] 87540 105560 101424 94584 68413 ...
##  $ app            : num [1:100000] 12 25 12 13 12 3 1 9 2 3 ...
##  $ device         : num [1:100000] 1 1 1 1 1 1 1 1 2 1 ...
##  $ os             : num [1:100000] 13 17 19 13 1 17 17 25 22 19 ...
##  $ channel        : num [1:100000] 497 259 212 477 178 115 135 442 364
135 ...
##  $ click_time     : POSIXct[1:100000], format: "2017-11-07 09:30:38"
"2017-11-07 13:40:27" ...
##  $ attributed_time: POSIXct[1:100000], format: NA NA ...
##  $ is_attributed  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  - attr(*, "spec")=
##   .. 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()
##   .. )
```

ETAPA 5: Divisão dos dados em treino e teste

Como há muitas linhas no dataset e poucos dados são referentes ao “is_attributed” == 0, para ter um dataset mais equilibrado, será inserido em um novo dataset(dadosT) apenas os dados em que “is_attributed” for igual a 1. Posteriormente, será retirada uma amostra de mesmo tamanho com “is_attributed” == 0 e salva em um novo dataset(dadosF).Os dois novos datasets serão agrupados, formando um novo dataset equilibrado para fazer a divisão (dados2):

Após criado um novo dataset, o mesmo será dividido em dados de treino e teste:

ETAPA 6: Treinamento do modelo O modelo será treinado utilizando o algoritmo de árvode de decisão:

ETAPA 7: Avaliação do modelo

```
##
## Call:
##  randomForest(formula = is_attributed ~ ip + app + device + os +
channel + click_time, data = dados_treino, ntree = 100, nodesize = 10)
##                Type of random forest: classification
```

```
##                       Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 9.75%
## Confusion matrix:
##     0   1 class.error
## 0 151   8  0.05031447
## 1  23 136  0.14465409
```

Aplicando o modelo sobre os dados de teste:

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
18  19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   0   1
1   1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37
38  39  40
##   1   1   0   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0   0   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57
58  59  60
##   1   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1
1   1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77
78  79  80
##   1   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0
0   0   0
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97
98  99 100
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119 120
##   0   0   1   0   0   0   0   0   0   0   0   0   1   0   0   0   0
0   0   0
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
##   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
## Levels: 0 1
```

Analisando a Confusion Matrix:

```
##    previsao
##      0  1
##   0 64  4
##   1  5 63

## Confusion Matrix and Statistics
##
##    previsao
##      0  1
##   0 64  4
```

```
##   1  5 63
##
##                Accuracy : 0.9338
##                  95% CI : (0.8781, 0.9693)
##     No Information Rate : 0.5074
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8676
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9403
##             Specificity : 0.9275
##          Pos Pred Value : 0.9265
##          Neg Pred Value : 0.9412
##              Prevalence : 0.4926
##          Detection Rate : 0.4632
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9339
##
##        'Positive' Class : 1
##
```

ETAPA 8: Otimização do modelo Como o modelo atingiu uma acurácia de 91%, não será feita nenhuma otimização, pois este percentual já é considerado ótimo.