Projeto 'Detecção de Fraudes no Tráfego de Cliques em Propagandas de Aplicações Mobile'

1. **Escopo**

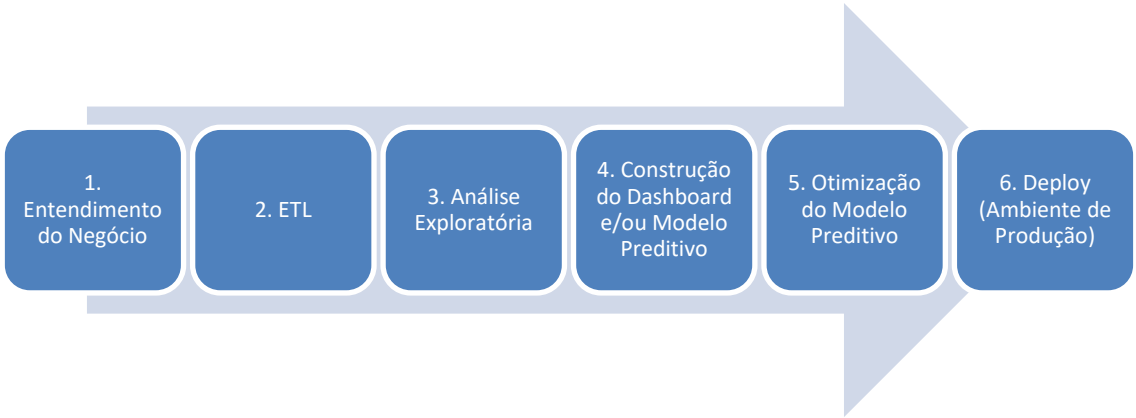
O risco de fraude está em toda parte, mas para as empresas que anunciam online, a fraude de cliques pode acontecer em um volume avassalador, resultando em dados de cliques enganosos e dinheiro desperdiçado. Os canais de anúncios podem aumentar os custos simplesmente quando pessoas ou bots clicam nos anúncios em grande escala, o que na prática não gera o resultado esperado. Com mais de 1 bilhão de dispositivos móveis em uso todos os meses, a China é o maior mercado móvel do mundo e, portanto, sofre com grandes volumes de tráfego fraudulento.

A TalkingData (https://www.talkingdata.com), a maior plataforma de Big Data independente da China, cobre mais de 70% dos dispositivos móveis ativos em todo o país. Eles lidam com 3 bilhões de cliques por dia, dos quais 90% são potencialmente fraudulentos. Sua abordagem atual para impedir fraudes de cliques para desenvolvedores de aplicativos é medir a jornada do clique de um usuário em todo o portfólio e sinalizar endereços IP que produzem muitos cliques, mas nunca acabam instalando aplicativos. Com essas informações, eles criaram uma lista negra de IPs e uma lista negra de dispositivos.

Embora bem-sucedidos, eles querem estar sempre um passo à frente dos fraudadores e pediram a sua ajuda para desenvolver ainda mais a solução. Você está desafiado a criar um algoritmo que possa prever se um usuário fará o download de um aplicativo depois de clicar em um anúncio de aplicativo para dispositivos móveis. Em resumo, neste projeto, você deverá construir um modelo de aprendizado de máquina para determinar se um clique é fraudulento ou não. Para a construção desse projeto, recomendamos a utilização da linguagem R e o dataset disponível no Kaggle em: https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data

2. **Proposta de Solução.**

Usamos a linguagem R, embora pudessem usar qualquer ferramenta proprietária ou gratuita para construir a solução, visto que o processo da Ciência de Dados é o mesmo independente de tecnologia. Dividimos o script em seis partes principais, no qual cada passo representa uma etapa do processo, veja:



4.1 Entendimento do Negócio

1. Objetivo

Prever se o usuário fará o download de um aplicativo ou não após clicar em um anúncio. Em suma, prever a ocorrência de fraudes (cliques que não confirmam em download de aplicativos, sejam pessoas ou robôs simulando o clique).

2. Dicionário de Dados

O dicionário está na pagina (na aba 'Data') do Kaggle, conforme mostrado acima. Veja:

Data fields

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.

The test data is similar, with the following differences:

- `click_id` : reference for making predictions
- `is_attributed` : not included

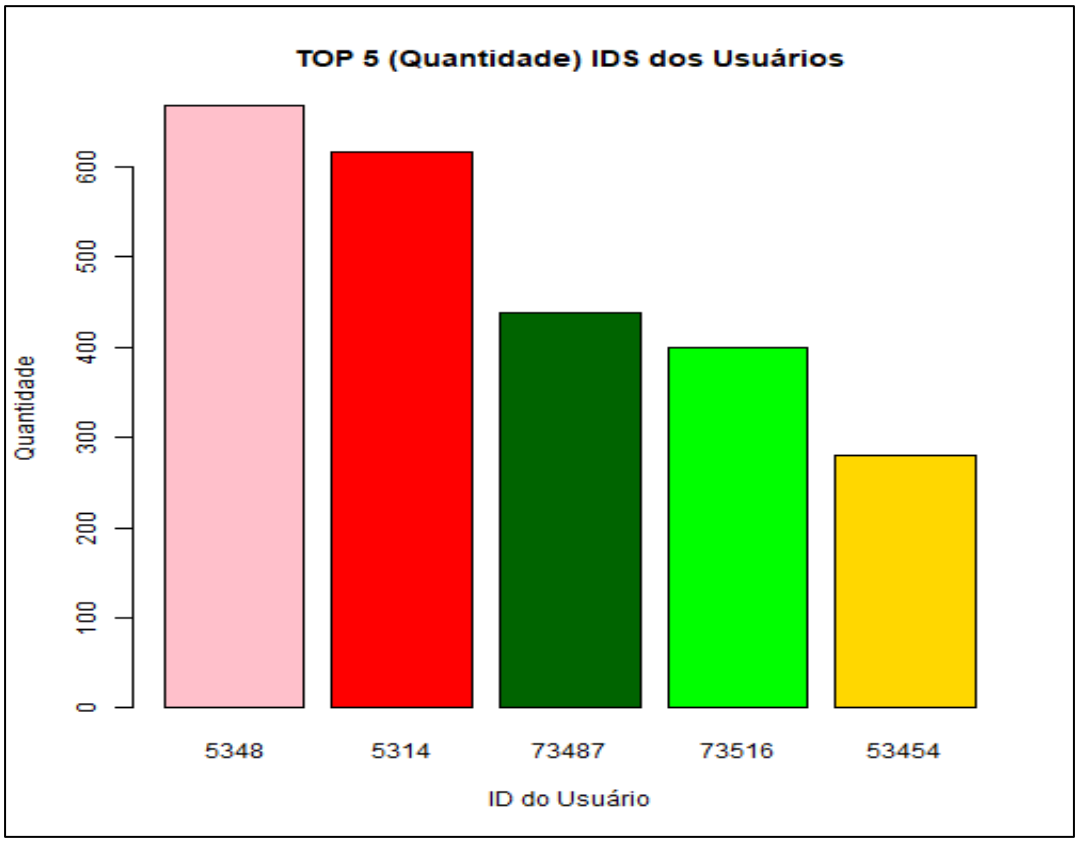
4.2 ETL

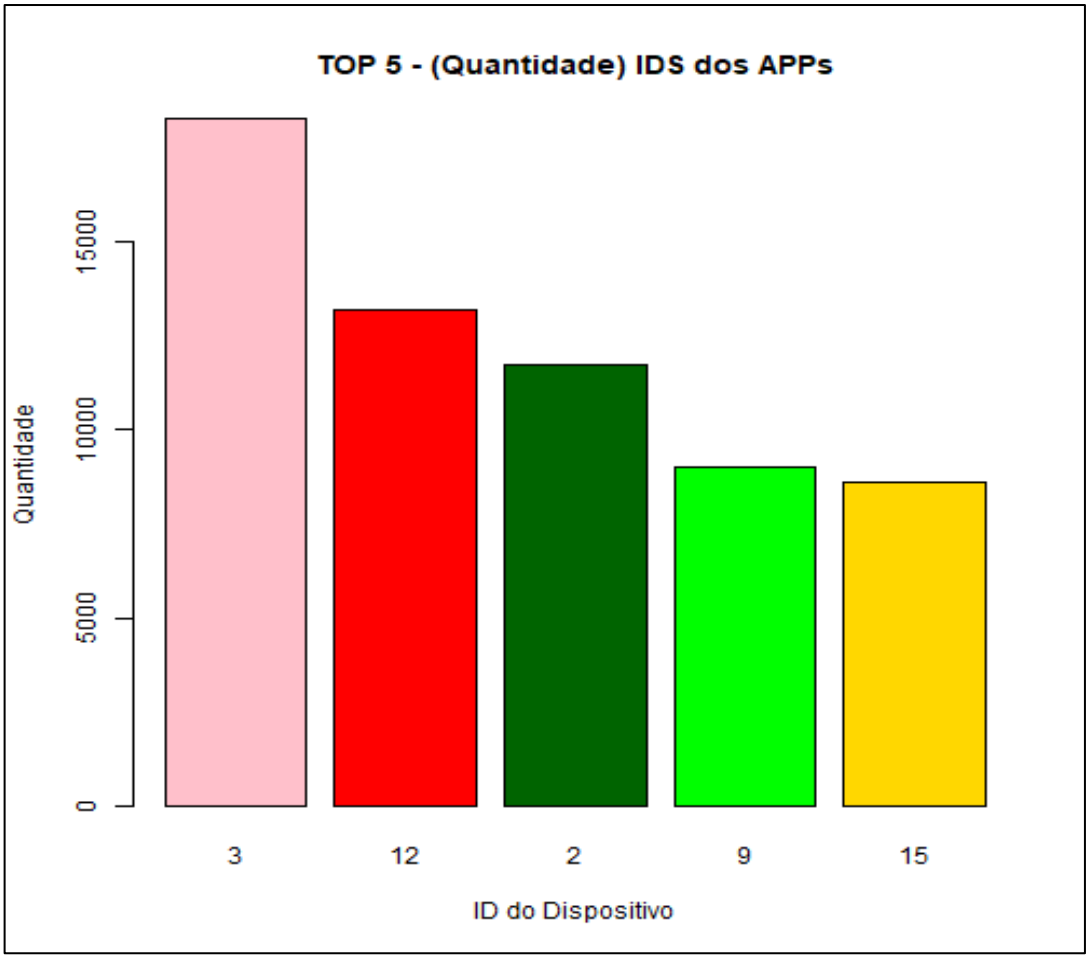
Optamos por coletar uma amostra (chamada train_sample.csv) do dataset original (train.csv) devido as limitações de processamento e armazenamento em nossa máquina, com um total de registros de 100.000. Algumas variáveis tiveram que ser transformadas de numéricas para categóricas ou vice-versa.

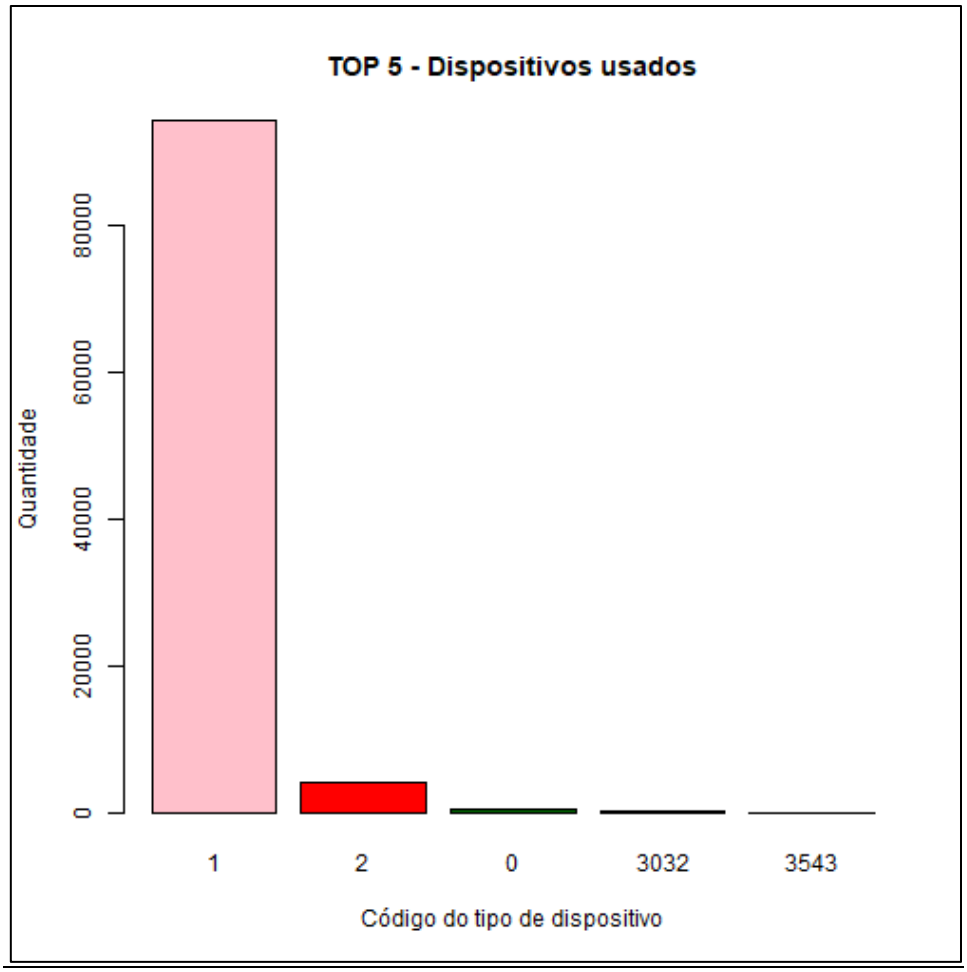
4.3 Análise Exploratória

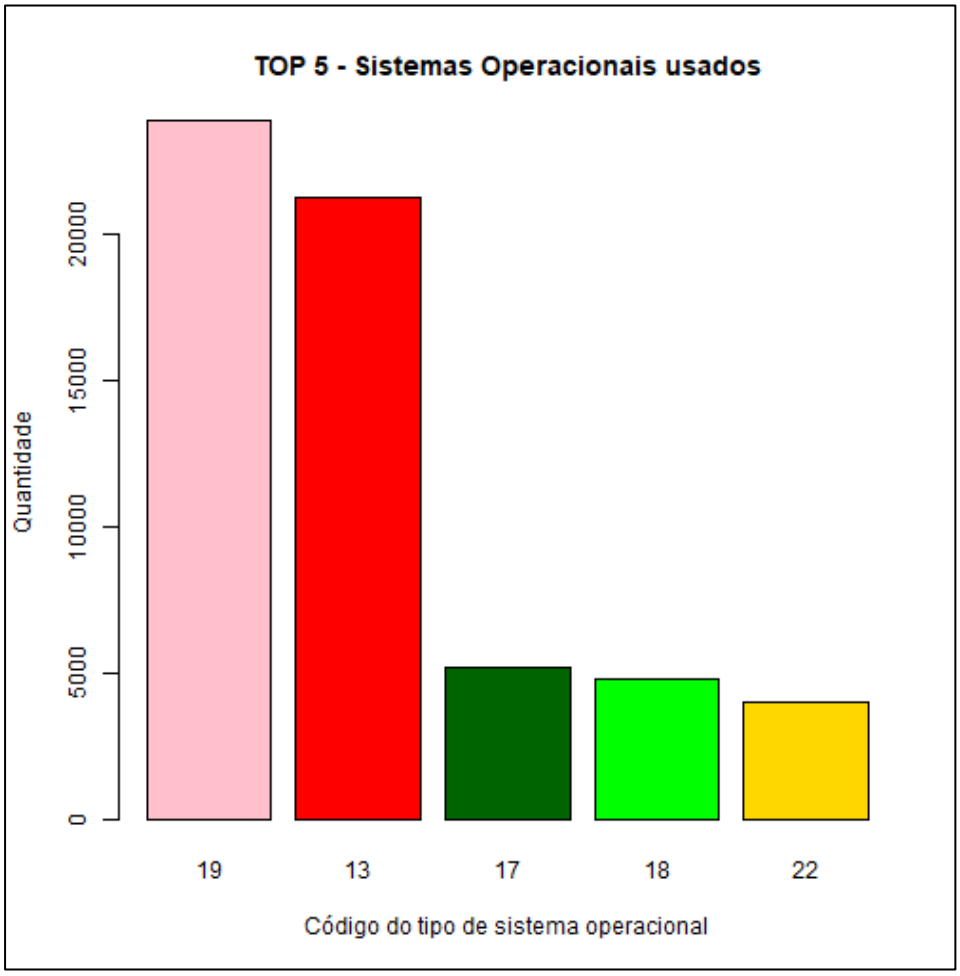
Na Análise Exploratória obtemos por meio de gráficos os seguintes insights abaixo. Importante ressaltar que não trabalhamos questões de design e ux nos gráficos visto que o propósito era serem gerados rapidamente e de forma interna (somente para os

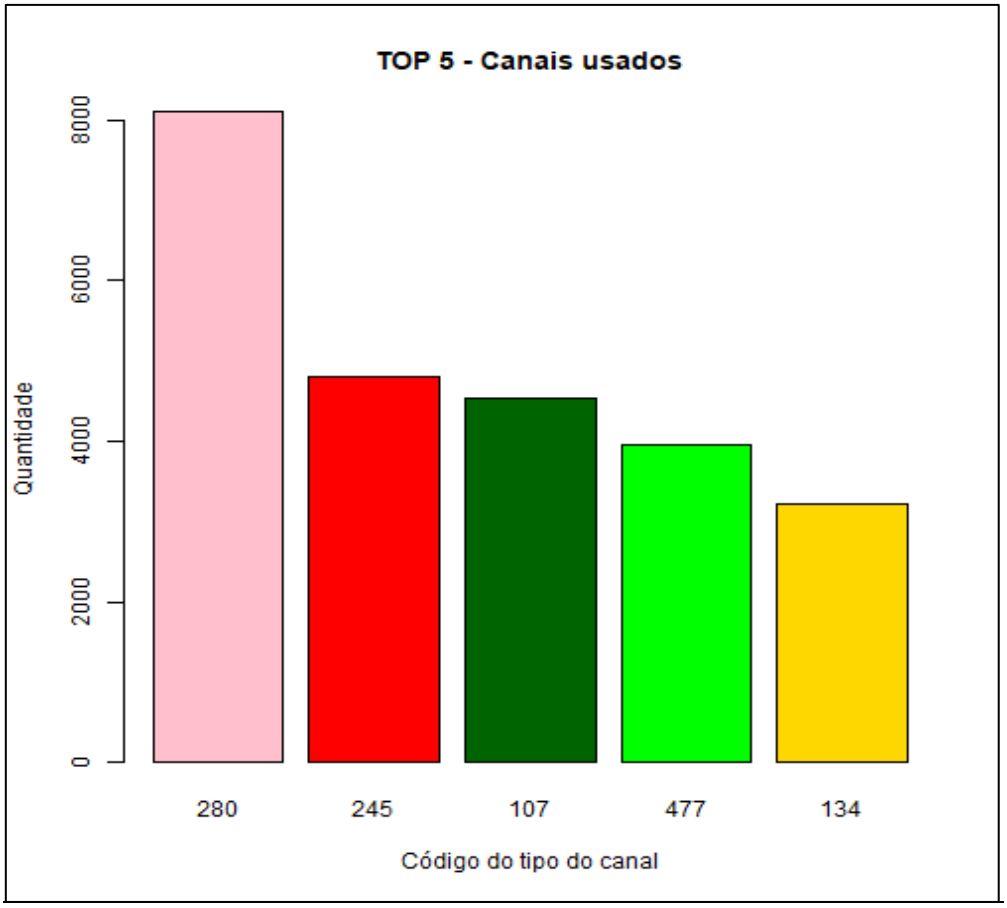
desenvolvedores). Se fosse construir um dashboard para o usuário final, certamente os gráficos sofreriam ajustes. Devido à quantidade grande de categorias no *datasets*, extraímos apenas os top 5 de algumas variáveis.

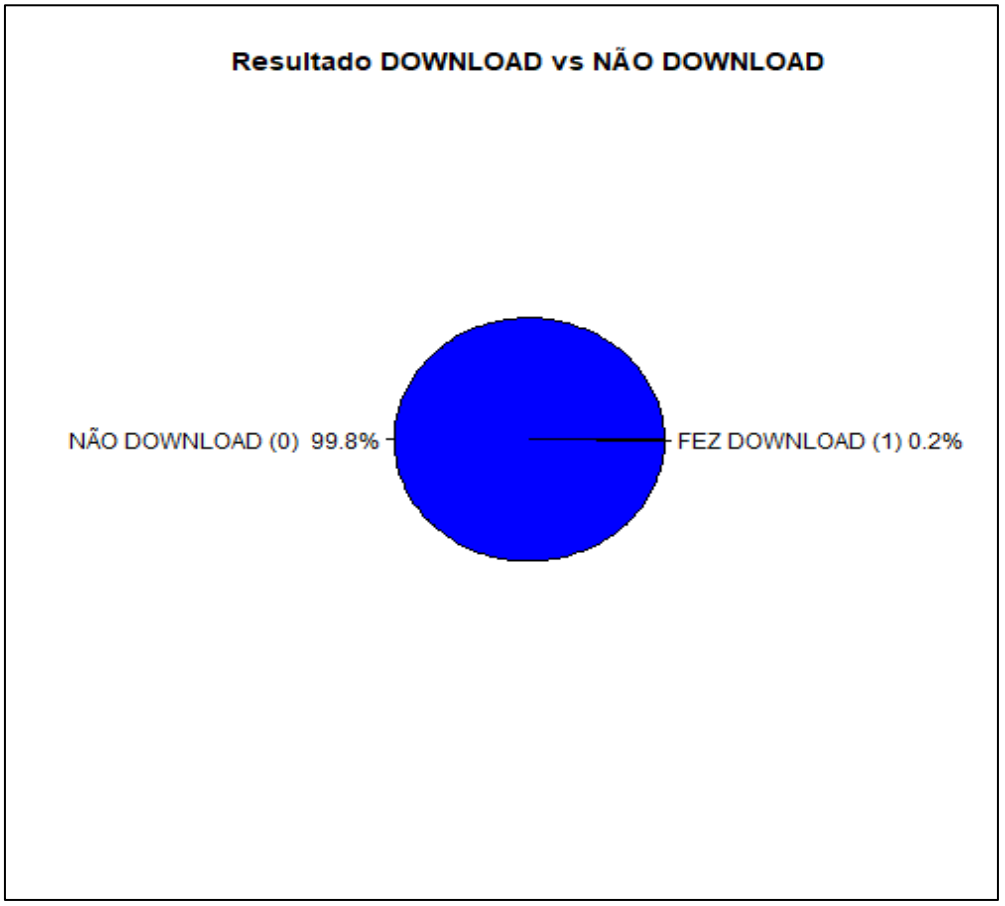


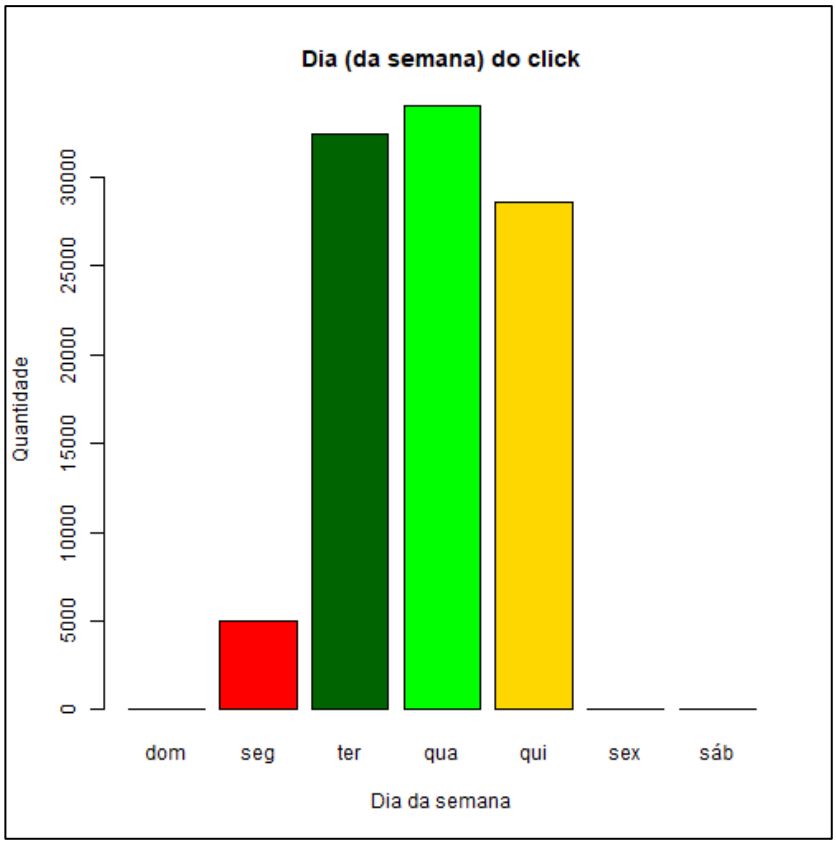


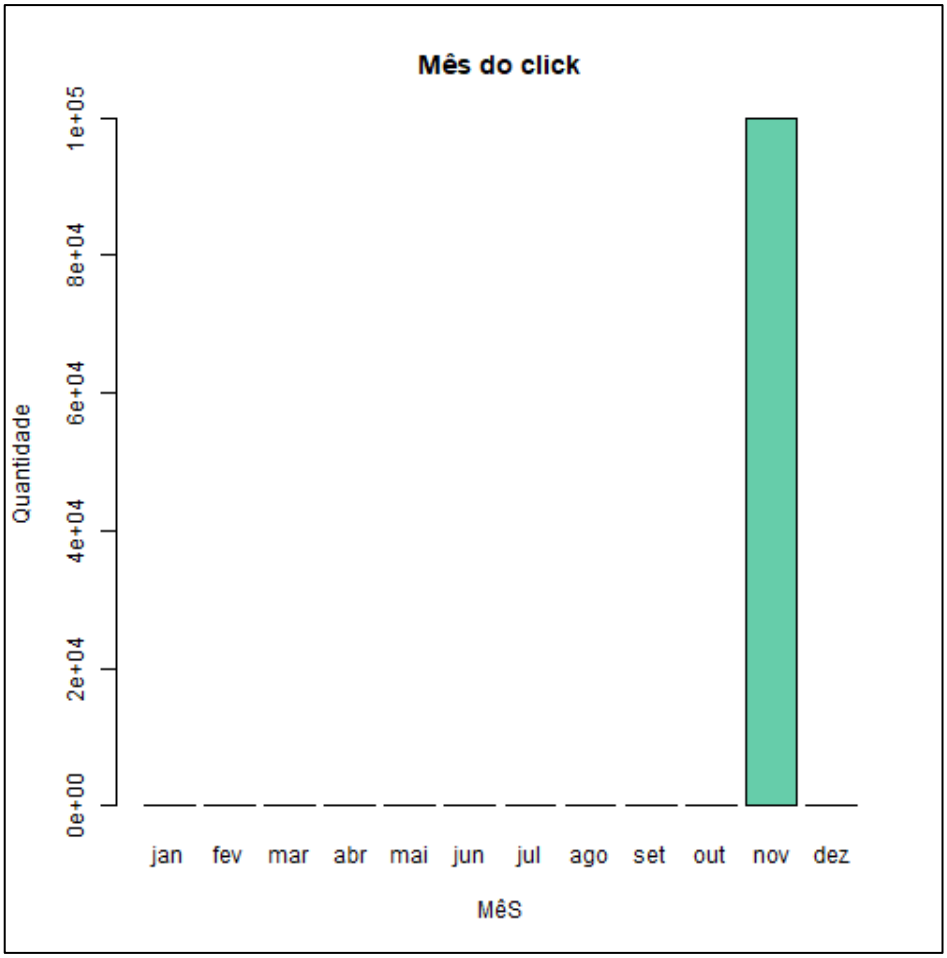

TOP 5 - Dispositivos usados
Quantidade
0
20000
40000
60000
80000
1
2
0
3032
3543
Código do tipo de dispositivo

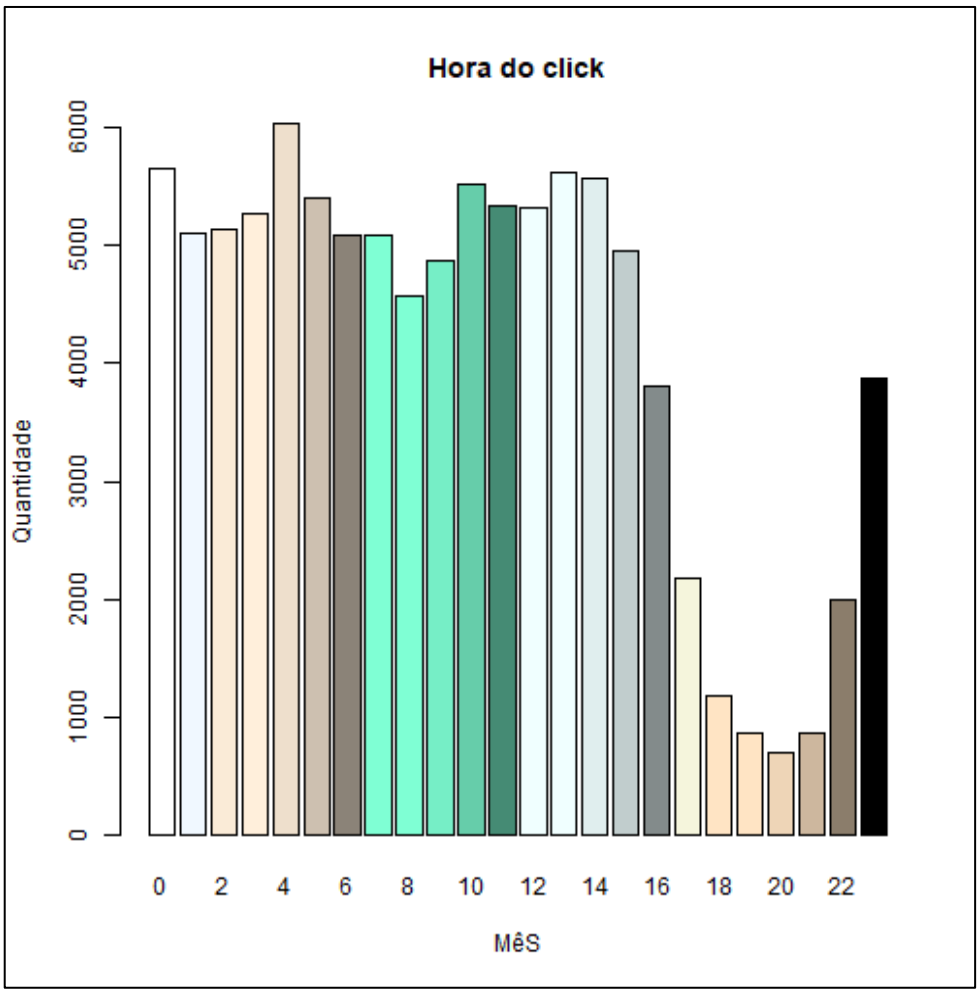

TOP 5 - Sistemas Operacionais usados
Quantidade
0
5000
10000
15000
20000
19
13
17
18
22
Código do tipo de sistema operacional

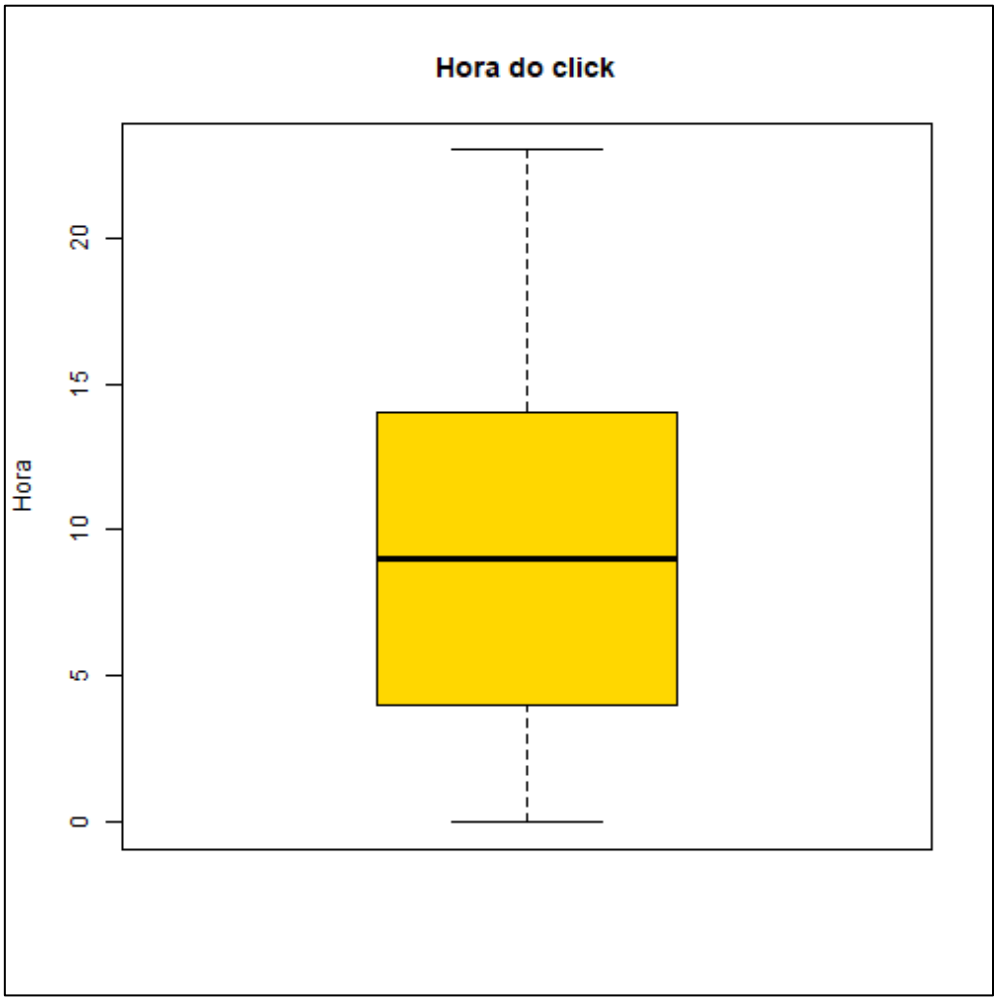

TOP 5 - Canais usados
Quantidade
0
2000
4000
6000
8000
280
245
107
477
134
Código do tipo do canal

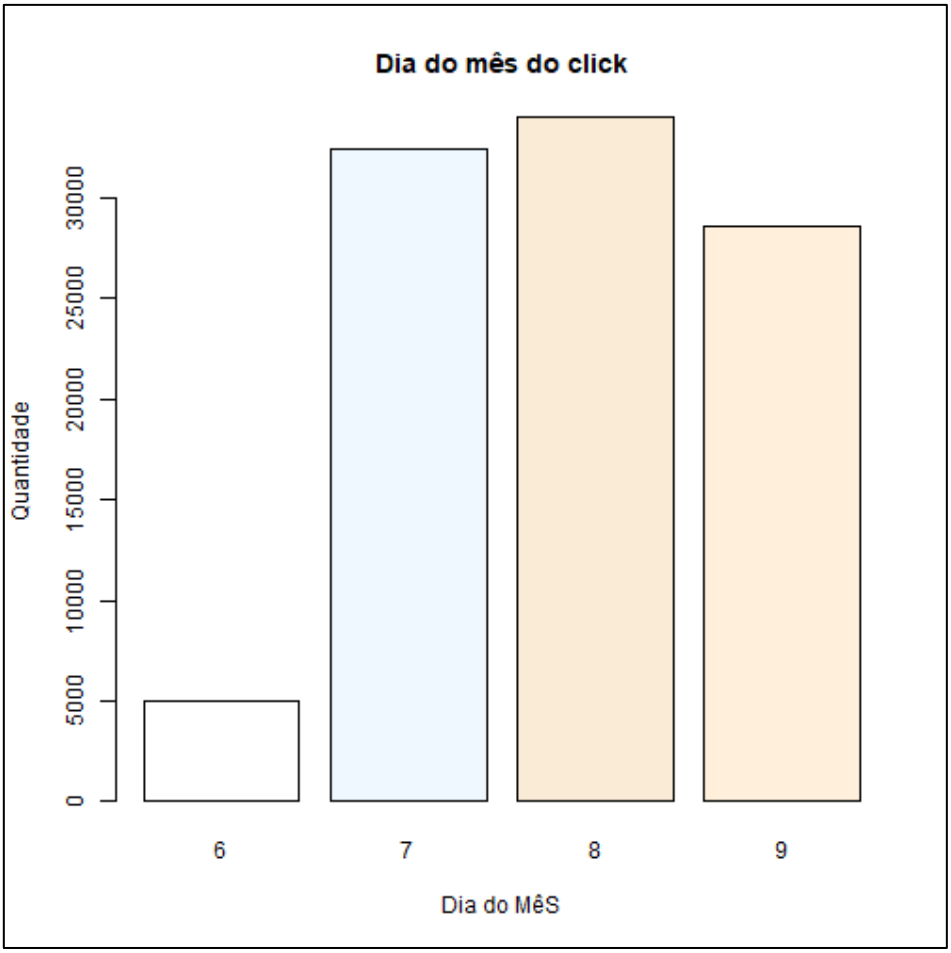

Resultado DOWNLOAD vs NÃO DOWNLOAD
NÃO DOWNLOAD (0) 99.8%
FEZ DOWNLOAD (1) 0.2%

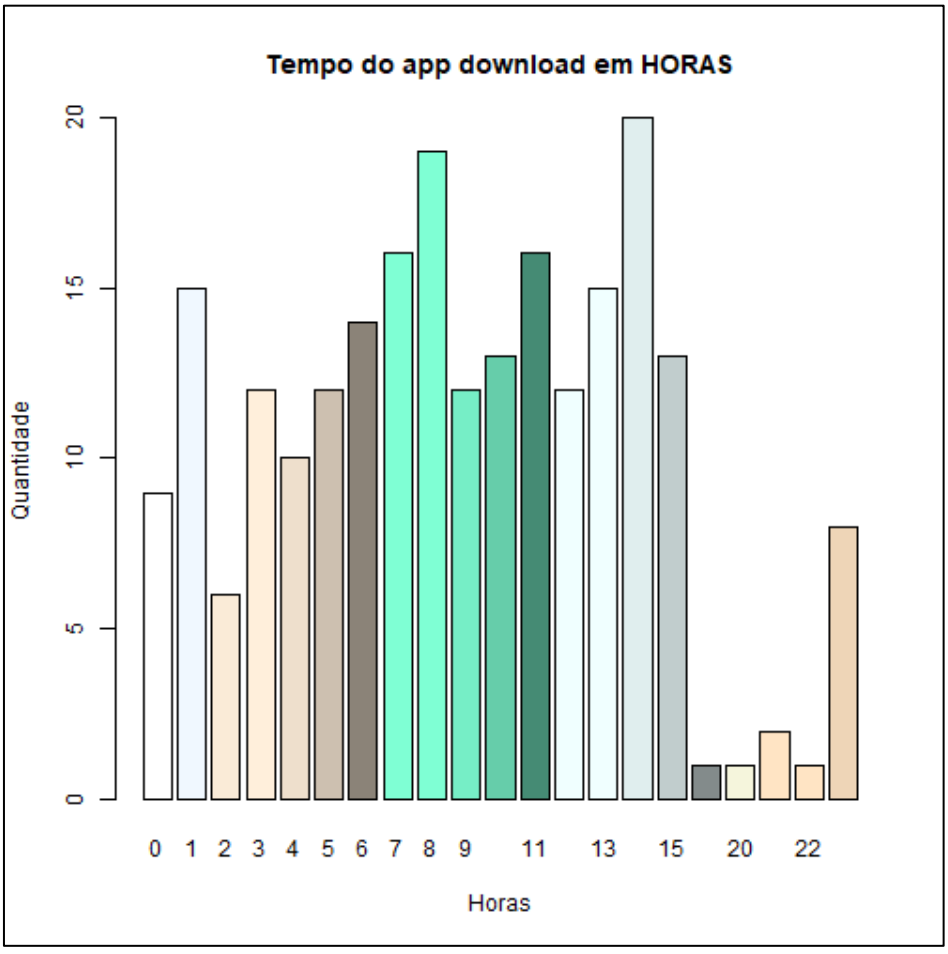

Dia (da semana) do click
Quantidade
0
5000
10000
15000
20000
25000
30000
dom
seg
ter
qua
qui
sex
sáb
Dia da semana

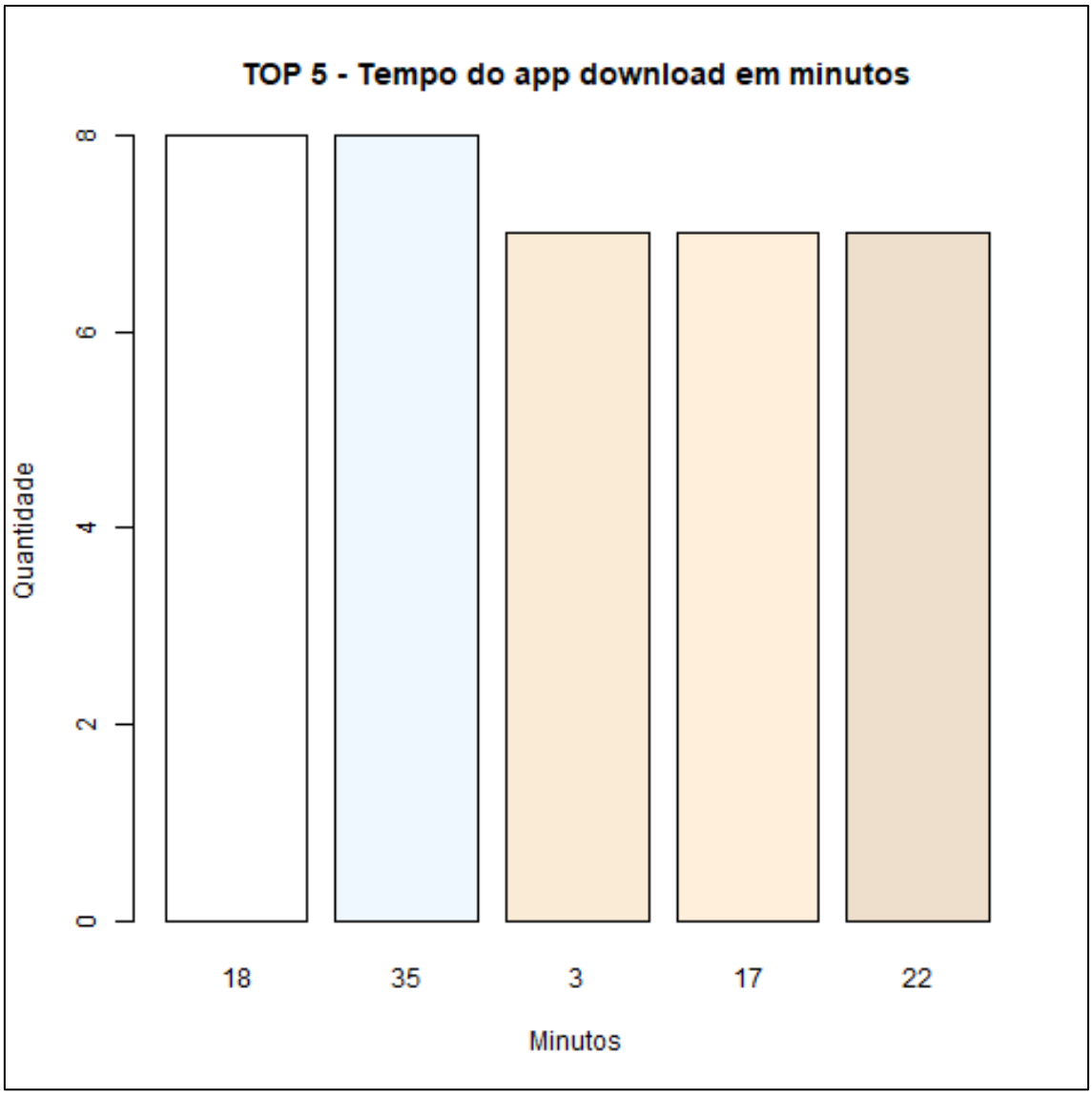

Mês do click
Quantidade
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
jan
fev
mar
abr
mai
jun
jul
ago
set
out
nov
dez
MêS


Hora do click
Quantidade
0
1000
2000
3000
4000
5000
6000
0
2
4
6
8
10
12
14
16
18
20
22
MêS


Hora do click
Hora
0
5
10
15
20


Dia do mês do click
Quantidade
0
5000
10000
15000
20000
25000
30000
6
7
8
9
Dia do MêS


Tempo do app download em HORAS
Quantidade
0
5
10
15
20
0 1 2 3 4 5 6 7 8 9 11 13 15 20 22
Horas



4.4 Construção do Modelo

Na construção do modelo preditivo (problema de classificação em questão), optamos em usar o 'modelo_rf' (pacote rpart), obtemos os seguintes resultados na confusition *matrix* abaixo, sendo que 1 representa 'FEZ DOWNLOAD' e 0 'NÃO FEZ DOWNLOAD' (indicando fraude). A métrica 'precisão' representa o número de acertos do modelo:

Modelo1		
Previsto/ real	0	1
0	29922	67
1	10	1
Precisão		99,7%
Erros		77

4.5 Otimização do Modelo

Construímos um segundo modelo, dessa vez usando o algoritmo 'naiveBayes' (pacote e1071) e retestamos para avaliar o modelo obtendo os seguintes resultados:

Modelo2		
Pred (previsto)	True (valor real)	Frequência
0	0	29563
1	0	369
0	1	20
1	1	48
Precisão		98,7%
Erros		389

Vimos que o resultado foi pior e o modelo 1 continua sendo o melhor. Removemos então a variável 'ip' e testamos novamente obtendo os seguintes resultados:

Modelo 3		
Pred (previsto)	True (valor real)	Frequência
0	0	29906
1	0	26
0	1	45
1	1	23
Precisão		99,7%
Erros		71

Aparentemente o modelo 3 teve um desempenho igual que o modelo 1 (99,7%), porém ao notar o número de erros é menor que o modelo 1. Logo o modelo 3 é o melhor. Resultado final:

Nome	Algoritmo	Precisão	Erros
Modelo 1	R-part	0,997633	77
Modelo 2	naiveBayes	0,987033	389
Modelo 3	R-parte com variável IP retirada	0,997633	71

6. Deploy

Escolhemos apenas optar por subir para o GITHUB. Outras alternativas: construir uma interface gráfica e disponibilizá-la para o cliente (por meio da nuvem).