Projeto 1 - Deteccao de Fraudes

Ana Paula Pancieri

2/16/2022

Detecção de Fraudes no Tráfego de Cliques em Propagandas de Aplicação Mobile

Este código foi criado para o projeto da Formação Cientista de Dados da Data Science Academy

Problema de Negócio: construir um modelo de aprendizado de máquina para determinar se um clique é fraudulento ou não.

As informações foram disponibilizado pela empresa TalkingData e podem ser encontradas no Kaggle https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/data

```
library (inspectdf)
library (tidyr)
library (readr)
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 (Amelia)
```

```
## Carregando pacotes exigidos: Rcpp
```

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.8.0, built: 2021-05-26)
## ## Copyright (C) 2005-2022 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
library (ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
library (caTools)
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(e1071)
library(caret)
```

```
## Carregando pacotes exigidos: lattice
```

```
library(rmarkdown)
```

Importando o dataset

Devido ao tamanho do dataset, as análises deste projeto foram realizadas utilizando apenas o dataset train_sample, que contém 100.000 amostras aleatórias do dataset principal. Carreguei apenas os dados de treino pois os dados de teste não possuem a target

```
df <- read_csv("train_sample.csv", show_col_types = FALSE)

str(df)
```

```
## spec_tbl_df [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()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

PREPARAÇÃO DOS DADOS

Processo de limpeza e análise inicial dos dados

```
# Criação das colunas de dia, hora
df$day <- format(df$click_time,"%d")
df$hour <- format(df$click_time,"%H")
```

```
# Criação da coluna com periodo do dia
df$shift <- format(df$click_time,"%H")

shift_day <-  function(i){
  if (i >= 0 & i < 6)
    return ("MADRUGADA")
  else if (i >= 6 & i <= 12)
    return ("MANHA")
  else if (i > 12 & i <= 18)
    return ("TARDE")
  else if (i > 18 & i <= 23)
    return ("NOITE")
}

df$shift <- sapply(as.integer(df$shift), shift_day)

# Transformação das variáveis para o tipo Fator
df$is_attributed <- as.factor(df$is_attributed)
df$shift <- as.factor(df$shift)

head(df)
```

```
## # A tibble: 6 x 11
##       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 4 more variables: is_attributed <fct>, day <chr>, hour <chr>,
## #   shift <fct>
```

ANÁLISE EXPLORATÓRIA

Breve análise exploratória do dataset df

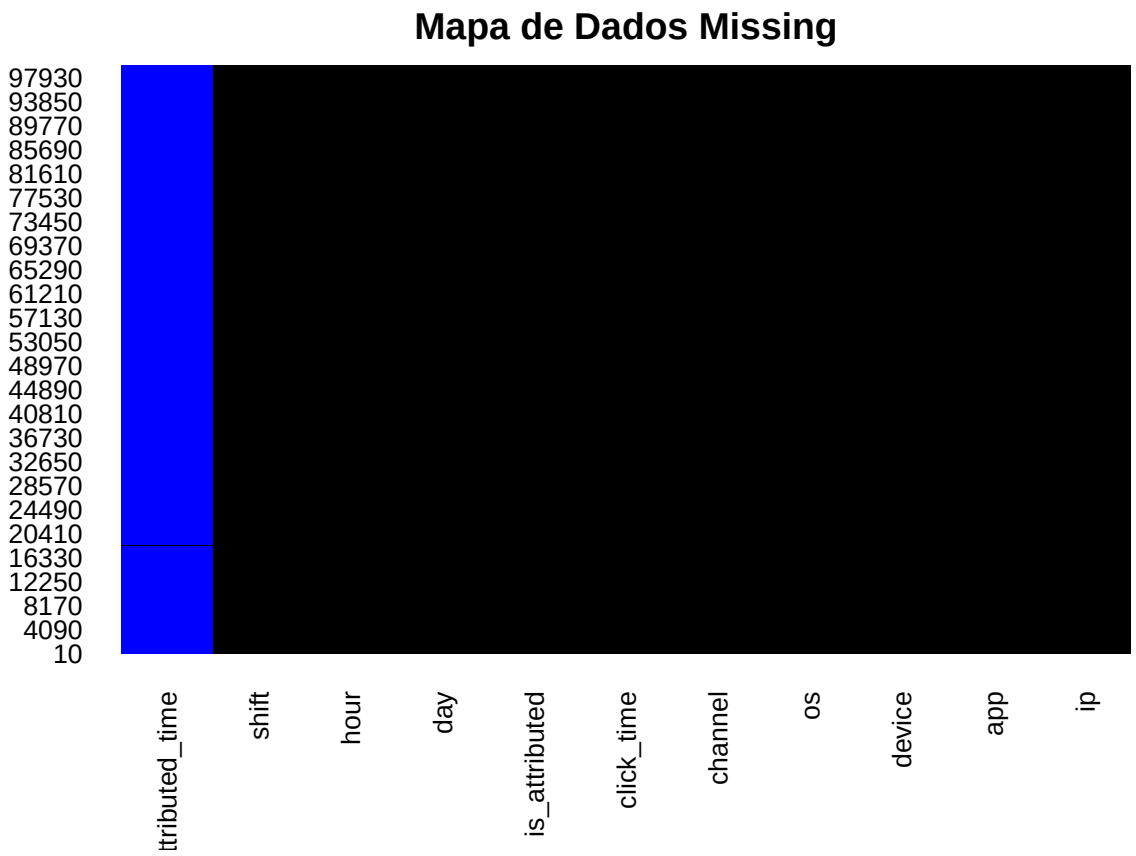
#Verificando os campos NA no dataset. A variável attributed_time possui 99.78% de dados NA que correspondem aos dados 0 da nossa variável target logo, essa variável será descartada no processo de modelagem.

```
missmap(df,
        main = "Mapa de Dados Missing",
        col = c("blue", "black"),
        legend = FALSE)
```

```
## Warning: Unknown or uninitialised column: `arguments`.

## Warning: Unknown or uninitialised column: `arguments`.

## Warning: Unknown or uninitialised column: `imputations`.
```

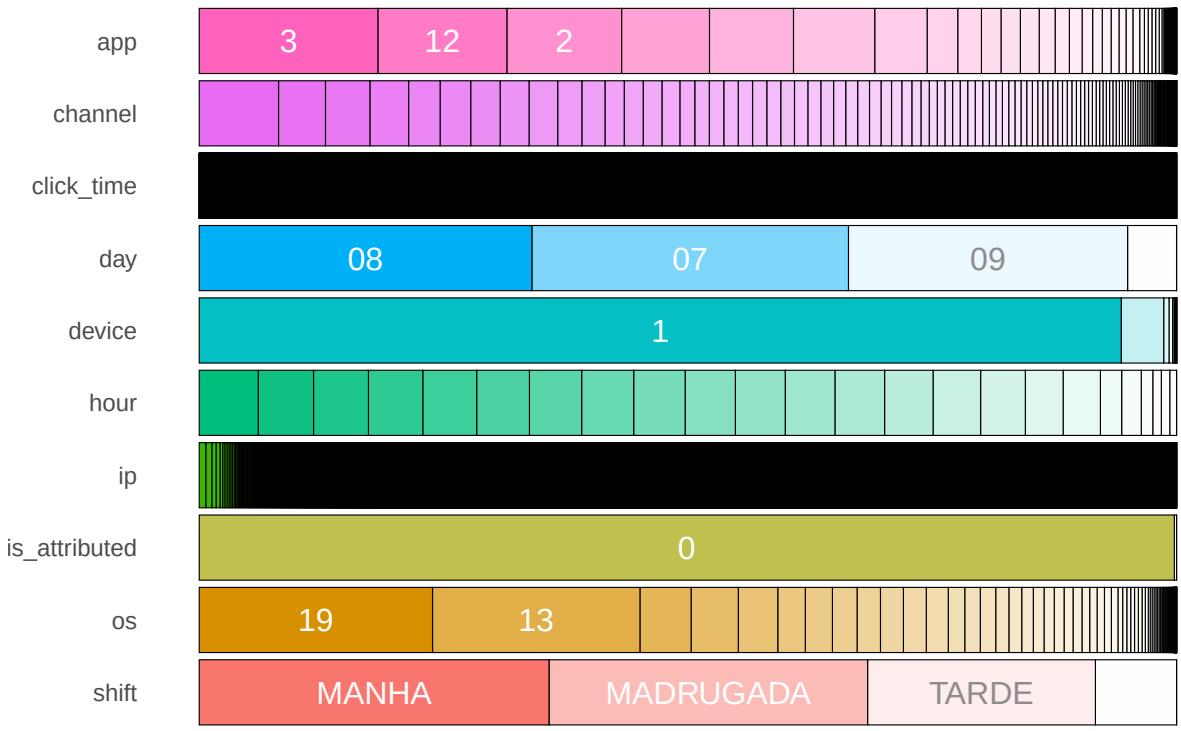



```
sapply(df, function(x) sum(is.na(x)/length(x))*100)
```

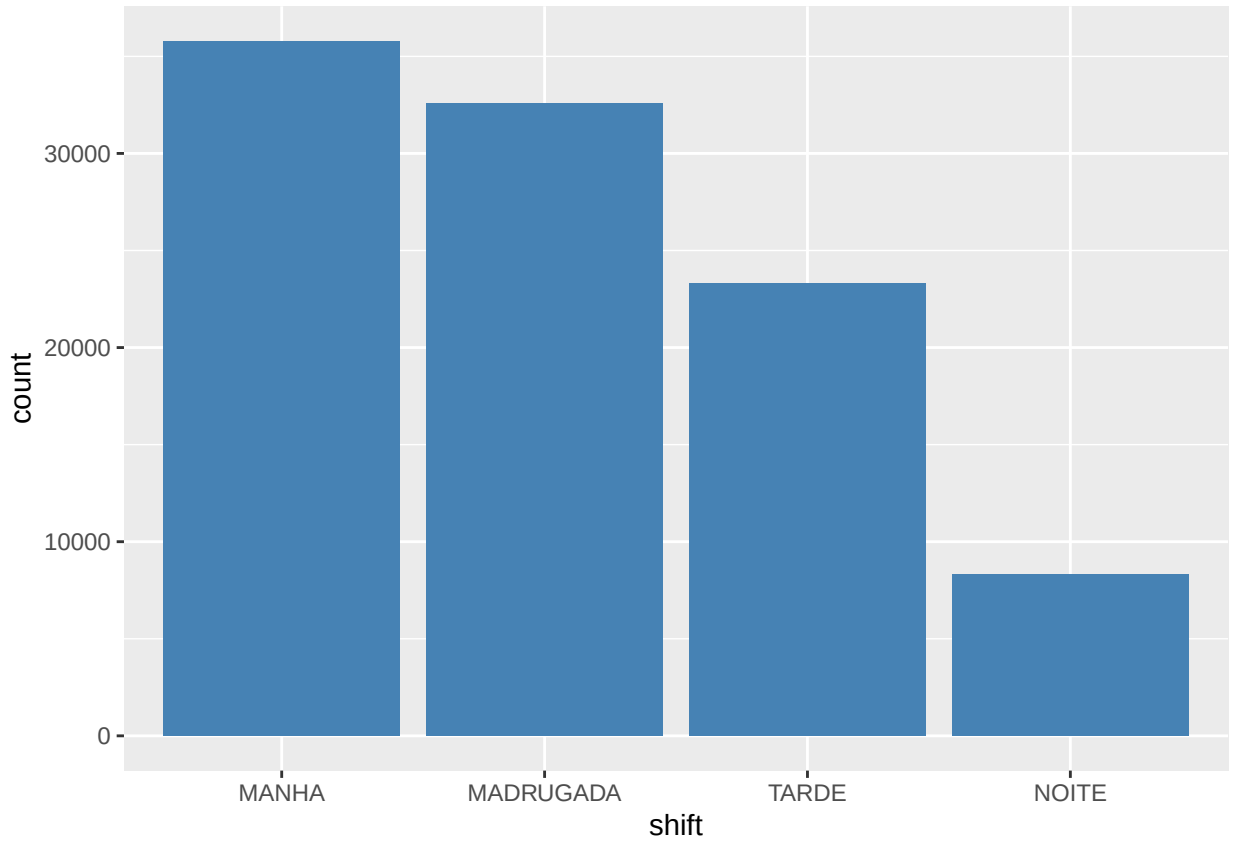
```
##             ip            app         device             os        channel
##          0.000          0.000          0.000          0.000          0.000
##     click_time attributed_time   is_attributed            day           hour
##          0.000         99.773          0.000          0.000          0.000
##          shift
##          0.000
```

Verificando a distribuição de todas as features observamos que apesar de algumas variáveis serem classificadas como numéricas, elas são fatores

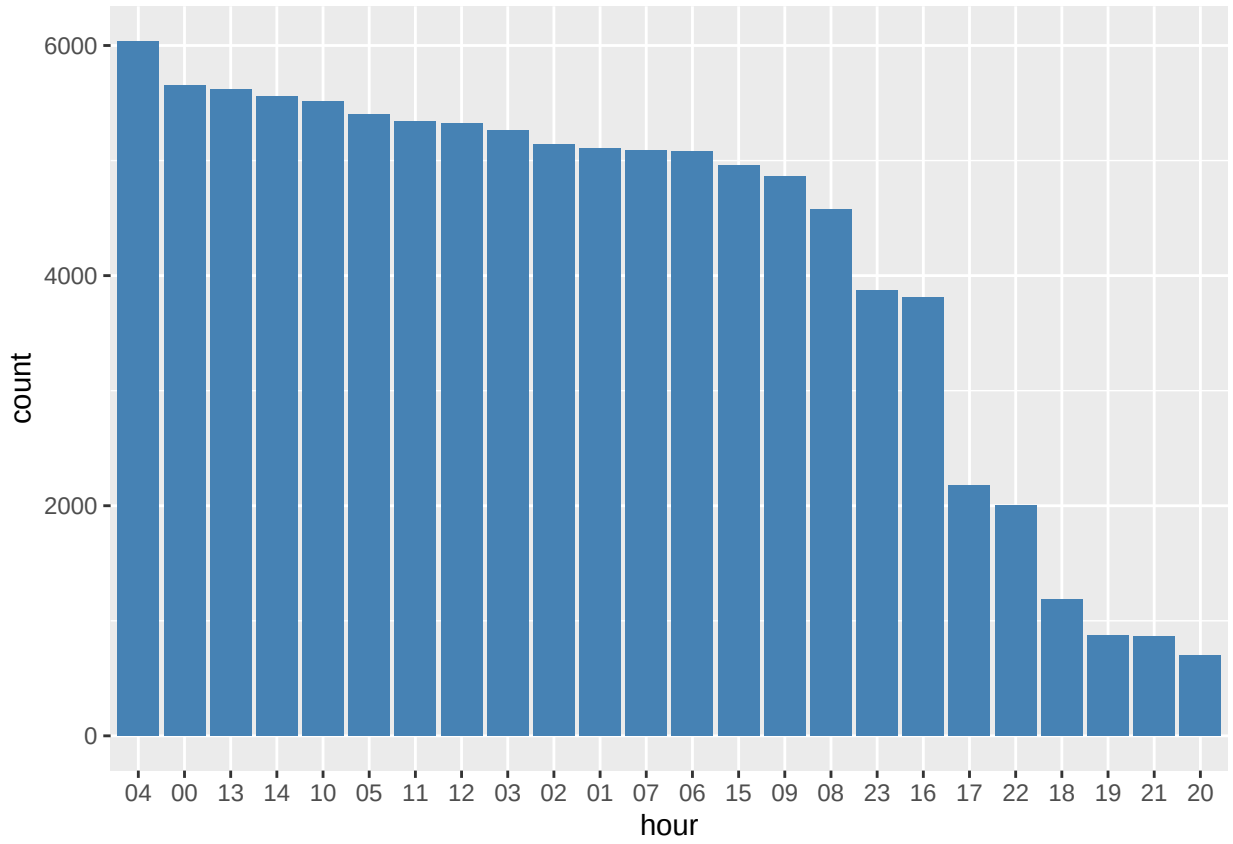
```
df %>%
  select(-attributed_time)%>%
  mutate_all(as.factor) %>%
  inspect_cat() %>%
  show_plot()
```



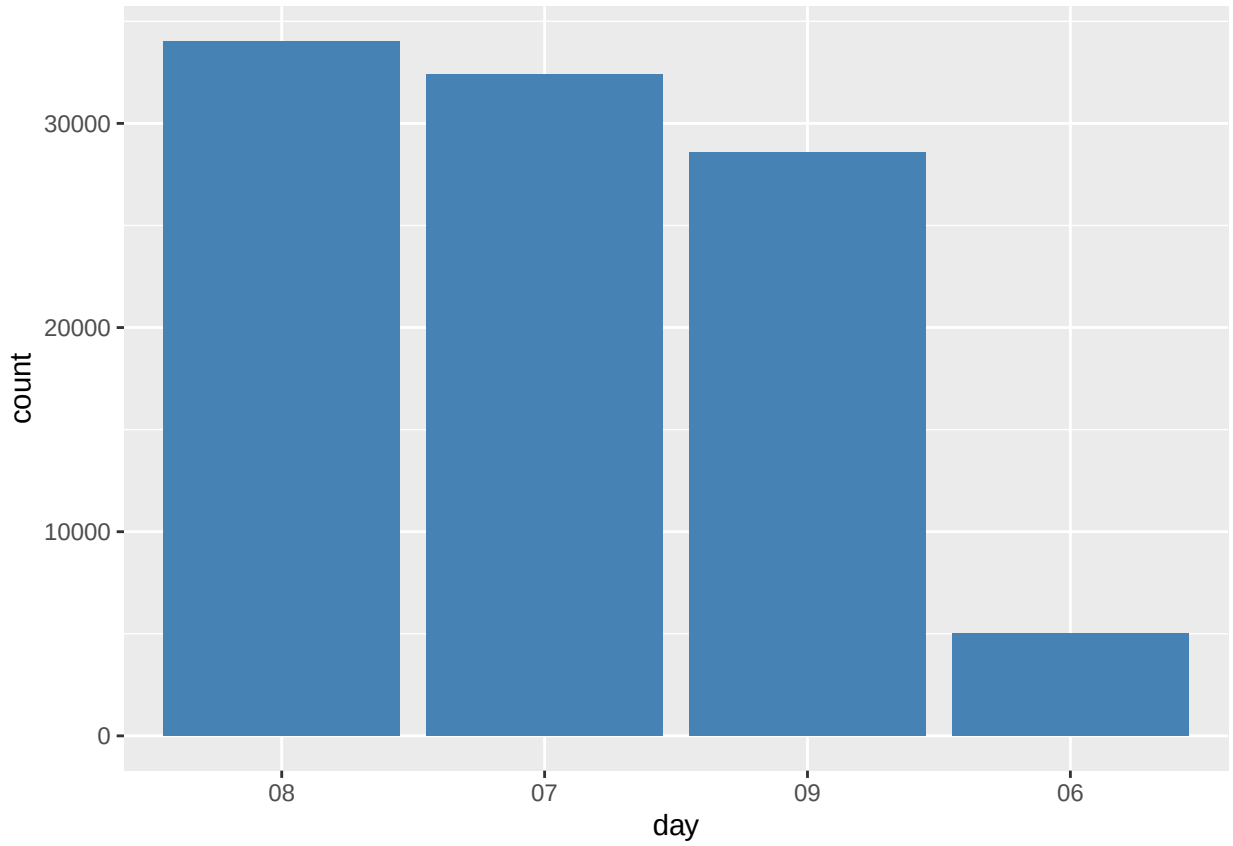

```
df %>%
ggplot(aes(x=reorder(shift, shift, function(x)-length(x)))) +
  geom_bar(fill='steelblue') +
  labs(x='shift')
```



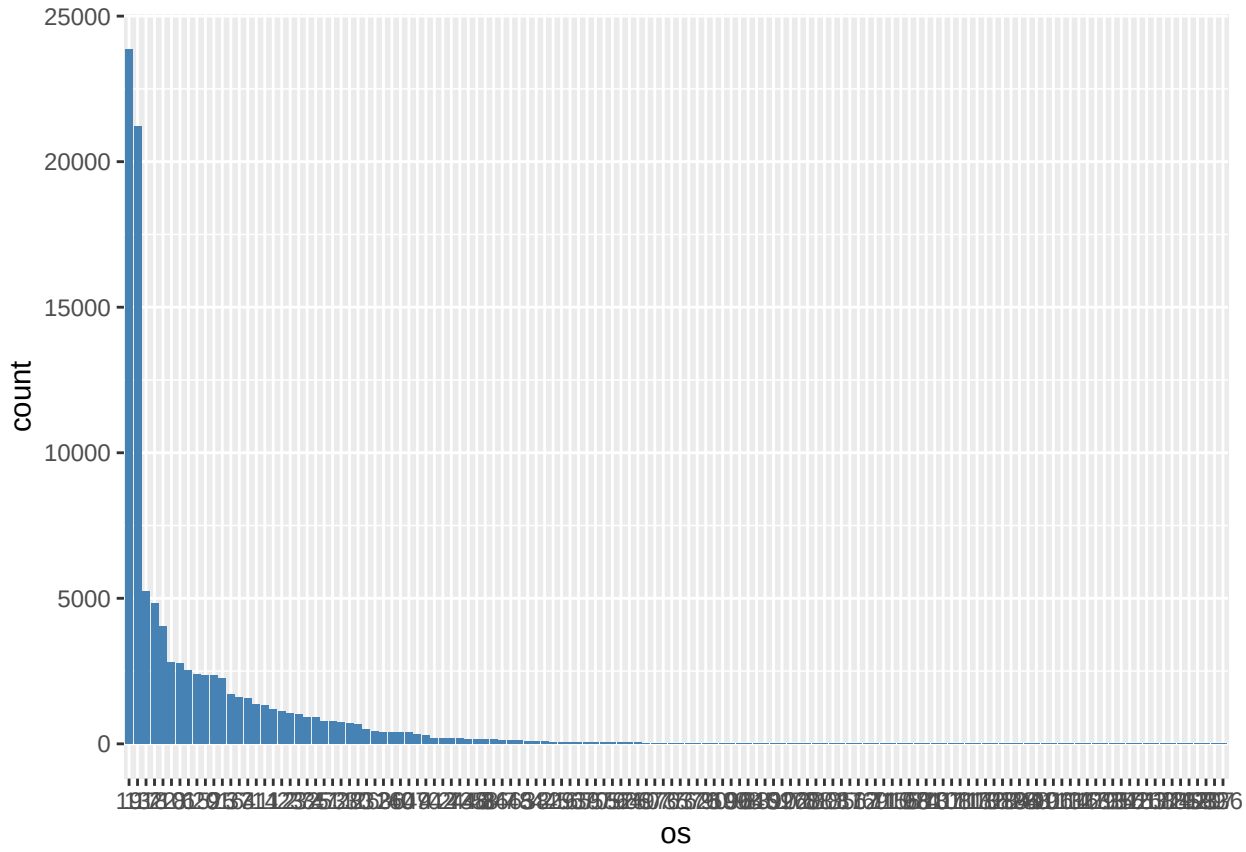

```
df %>%
ggplot(aes(x=reorder(hour, hour, function(x)-length(x)))) +
  geom_bar(fill='steelblue') +
  labs(x='hour')
```



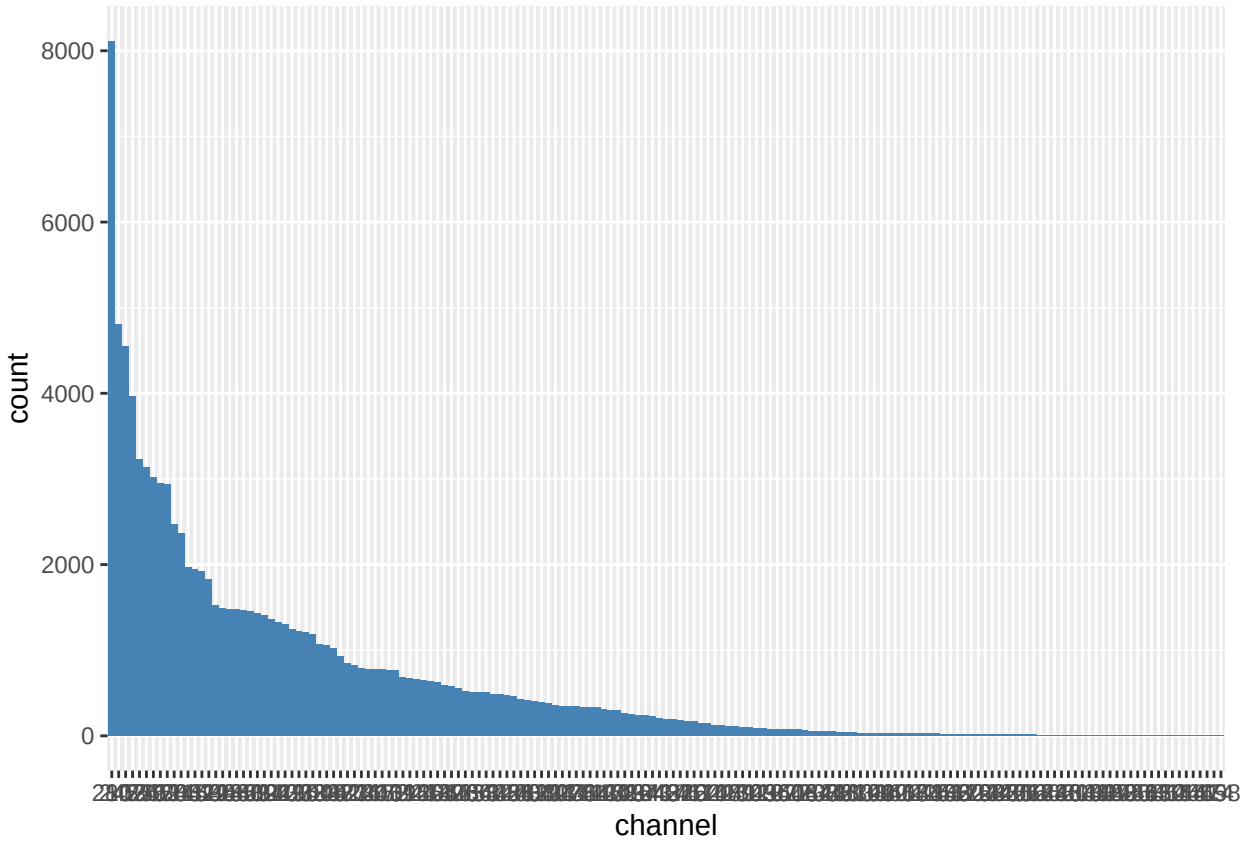

```
df %>%
  ggplot(aes(x=reorder(day, day, function(x)-length(x)))) +
  geom_bar(fill='steelblue') +
  labs(x='day')
```




```
df %>%
ggplot(aes(x=reorder(os, os, function(x)-length(x)))) +
  geom_bar(fill='steelblue') +
  labs(x='os')
```

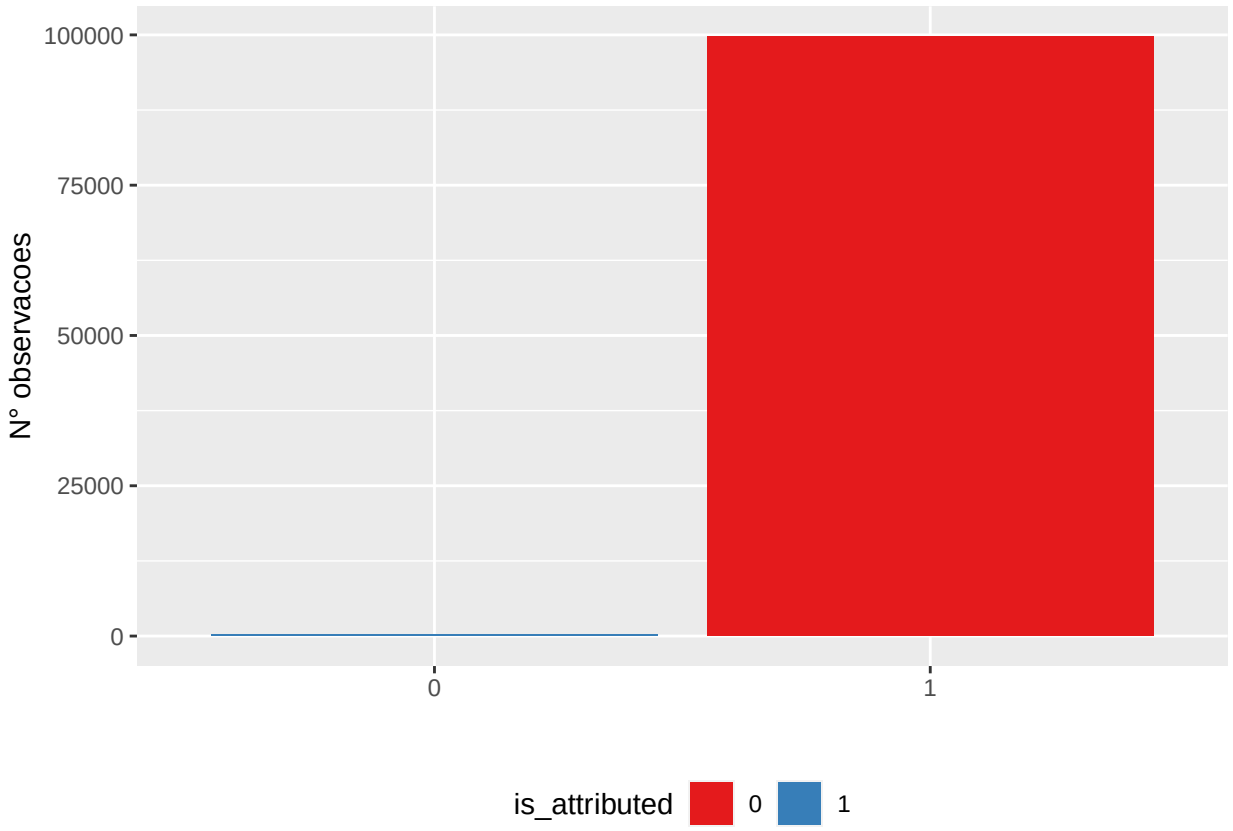



```
df %>%
ggplot(aes(x=reorder(channel, channel, function(x)-length(x)))) +
  geom_bar(fill='steelblue') +
  labs(x='channel')
```



A distribuição dos dados da variável Target mostra um grande desbalanceamento

```
df %>%
  count(is_attributed) %>%
  ggplot(aes(x=rev(is_attributed), y=n, fill=is_attributed))+
  geom_bar(stat = "identity")+
  scale_fill_brewer(palette="Set1")+
  theme(legend.position = "bottom")+
  labs(y="N° observacoes", x = "")
```




```
prop <- table(df$is_attributed)
prop <- prop.table(prop) * 100
round(prop, digits = 1)
```

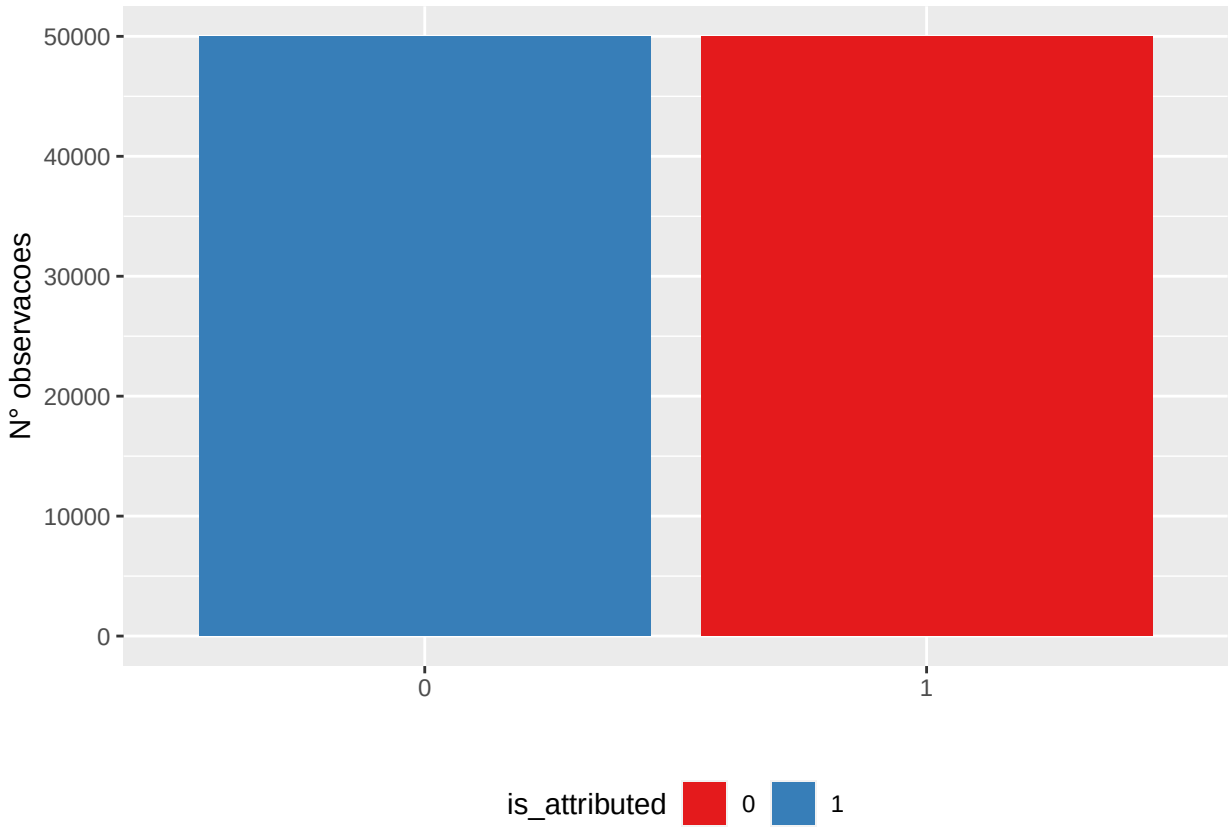
```
##
##    0    1
## 99.8  0.2
```

BALANCEANDO OS DADOS

Antes de fazer o balanceamento, criei novo dataset excluindo as 2 variáveis de tempo

```
df_1 <- df %>% select(-attributed_time, -click_time)
df.balanced <- ovun.sample(is_attributed ~ ., data=df_1,
                           N=nrow(df_1), p=0.5,
                           seed=1, method="both")$data

df.balanced %>%
  count(is_attributed) %>%
  ggplot(aes(x=rev(is_attributed), y=n, fill=is_attributed))+
  geom_bar(stat = "identity")+
  scale_fill_brewer(palette="Set1")+
  theme(legend.position = "bottom")+
  labs(y="N° observacoes", x = "")
```



TREINANDO O MODELO

Iniciando o treinamento - fazendo a divisão dos dados balanceados em 70/30

```
# Split dos dados
split = sample.split(df.balanced$is_attributed, SplitRatio = 0.70)

# Datasets de treino e de teste
trainset <- subset(df.balanced, split == TRUE)
testset <- subset(df.balanced, split == FALSE)
```

Modelo 1

Treinando o modelo com SVM utilizando todas as preditoras e fazendo previsões. Neste primeiro modelo tivemos acurácia de 87,87%

```
model_svm <- svm(is_attributed ~ .,
                 data = trainset,
                 type = 'C-classification',
                 kernel = 'radial')

# Previsões nos dados de treino
pred_train <- predict(model_svm, trainset)
mean(pred_train == trainset$is_attributed)
```

```
## [1] 0.8765286
```

```r
# Previsões nos dados de teste
pred_test <- predict(model_svm, testset)
mean(pred_test == testset$is_attributed)
```

```
## [1] 0.8786667
```

```r
# Matriz de Confusão
cm_svm <- confusionMatrix(table(pred = pred_test, testset$is_attributed),
                          positive = "1")
cm_svm
```

```
## Confusion Matrix and Statistics
##
##
## pred     0     1
##    0 13123  1761
##    1  1879 13237
##
##                Accuracy : 0.8787
##                  95% CI : (0.8749, 0.8823)
##     No Information Rate : 0.5001
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7573
##
##  Mcnemar's Test P-Value : 0.05247
##
##             Sensitivity : 0.8826
##             Specificity : 0.8748
##          Pos Pred Value : 0.8757
##          Neg Pred Value : 0.8817
##              Prevalence : 0.4999
##          Detection Rate : 0.4412
##    Detection Prevalence : 0.5039
##       Balanced Accuracy : 0.8787
##
##        'Positive' Class : 1
##
```

```r
# Métricas do Modelo
acuracia_svm <- cm_svm$overall["Accuracy"]
precisao_svm <- cm_svm$byClass["Precision"]
recall_svm <- cm_svm$byClass["Recall"]
f1_svm <- cm_svm$byClass["F1"]
metric_svm <- c(acuracia_svm, precisao_svm, recall_svm, f1_svm)
metric_svm
```

```
##  Accuracy Precision    Recall        F1
## 0.8786667 0.8756946 0.8825843 0.8791260
```

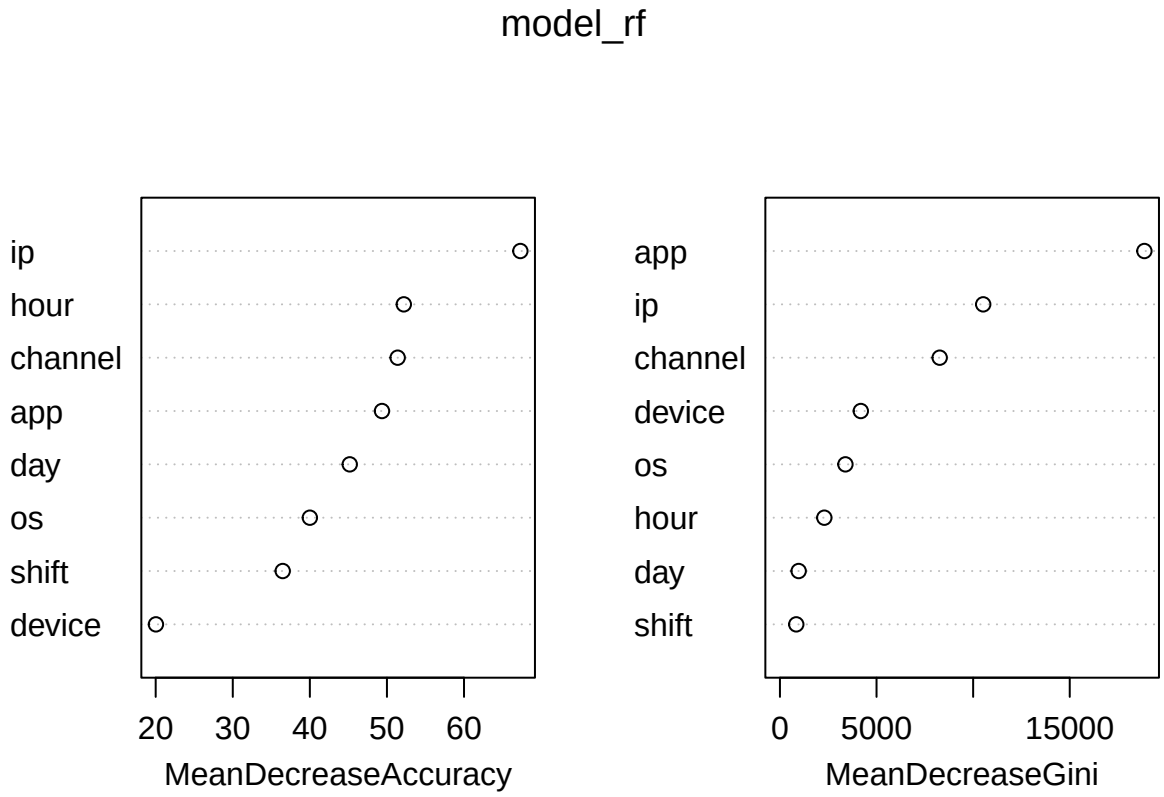
OTIMIZAÇÃO DO MODELO

Para buscar melhoria na acurácia do modelo, apliquei o Random Forest para selecionar variáveis.

```r
model_rf <- randomForest(is_attributed ~ .,
                         data = df.balanced,
```

```
                          ntree = 100, nodesize = 10, importance = T)

varImpPlot(model_rf)
```



Modelo 2

Treinei novamente o modelo usando SVM e as novas preditoras buscando melhoria e não encontrei diferença na acurácia, porém encontrei melhor precisão. A acurácia foi de 87,83%

```
# Treinando o modelo
model_svm2 <- svm(is_attributed ~ ip + channel + app + device,
                  data = trainset,
                  type = 'C-classification',
                  kernel = 'radial')

# Previsões nos dados de treino
pred_train2 <- predict(model_svm2, trainset)
mean(pred_train2 == trainset$is_attributed)
```

```
## [1] 0.8745143
```

```
# Previsões nos dados de teste
pred_test2 <- predict(model_svm2, testset)
mean(pred_test2 == testset$is_attributed)
```

```
## [1] 0.8783333
```

```
# Matriz de Confusão
mc_svm2 <- confusionMatrix(table(pred = pred_test2, testset$is_attributed),
```

```
                           positive = "1")
mc_svm2
```

```
## Confusion Matrix and Statistics
##
##
## pred     0     1
##    0 13498  2146
##    1  1504 12852
##
##                Accuracy : 0.8783
##                  95% CI : (0.8746, 0.882)
##     No Information Rate : 0.5001
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7567
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8569
##             Specificity : 0.8997
##          Pos Pred Value : 0.8952
##          Neg Pred Value : 0.8628
##              Prevalence : 0.4999
##          Detection Rate : 0.4284
##    Detection Prevalence : 0.4785
##       Balanced Accuracy : 0.8783
##
##        'Positive' Class : 1
##
```

```
# Métricas do Modelo
acuracia_svm2 <- mc_svm2$overall["Accuracy"]
precisao_svm2 <- mc_svm2$byClass["Precision"]
recall_svm2 <- mc_svm2$byClass["Recall"]
f1_svm2 <- mc_svm2$byClass["F1"]
metric_svm2 <- c(acuracia_svm2, precisao_svm2, recall_svm2, f1_svm2)
metric_svm2
```

```
##  Accuracy Precision    Recall        F1
## 0.8783333 0.8952354 0.8569143 0.8756558
```

Modelo 3

Uma vez que nas duas tentativas utilizando SVM os resultados foram similares, fiz nova tentativa, desta vez com o Random Forest para buscar melhorias nas métricas do modelo. A acurácia apresentada foi excelente, com 99.9% de acertos.

```
# Treinando o Modelo
model_forest <- randomForest(is_attributed ~ ip + channel + app + device + hour,
                             data = df.balanced,
                             ntree = 100, nodesize = 10)

# Previsões
pred_train_forest <- predict(model_forest, trainset)
```

```r
mean(pred_train_forest == trainset$is_attributed)
```

```
## [1] 0.9989
```

```r
pred_test_forest <- predict(model_forest, testset)
mean(pred_test_forest == testset$is_attributed)
```

```
## [1] 0.9990333
```

```r
# Matriz de Confusão
mc_forest <- confusionMatrix(table(pred = pred_test_forest, testset$is_attributed),
                             positive = "1")
mc_forest
```

```
## Confusion Matrix and Statistics
##
##
## pred     0     1
##    0 14973     0
##    1    29 14998
##
##                Accuracy : 0.999
##                  95% CI : (0.9986, 0.9994)
##     No Information Rate : 0.5001
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9981
##
##  Mcnemar's Test P-Value : 1.999e-07
##
##             Sensitivity : 1.0000
##             Specificity : 0.9981
##          Pos Pred Value : 0.9981
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4999
##          Detection Rate : 0.4999
##    Detection Prevalence : 0.5009
##       Balanced Accuracy : 0.9990
##
##        'Positive' Class : 1
##
```

```r
# Métricas do Modelo
acuracia_forest <- mc_forest$overall["Accuracy"]
precisao_forest <- mc_forest$byClass["Precision"]
recall_forest <- mc_forest$byClass["Recall"]
f1_forest <- mc_forest$byClass["F1"]
metric_forest <- c(acuracia_forest, precisao_forest, recall_forest, f1_forest)
metric_forest
```

```
##  Accuracy Precision    Recall        F1
## 0.9990333 0.9980701 1.0000000 0.9990341
```

CONCLUSÃO

Na busca de melhores resultados testei diferentes modelos e tentei otimizar a abordagem utilizada com diferentes variáveis preditoras e alcançando diferentes resultados.