Incognia Data Analysis

Hugo Daher

2022-03-02

Incognia Data Analysis Report

Introduction

Statistical analysis is a efficient way to get insights about any data, making anyone able to ask the right question. In addition, using the best technologies is crucial to get better results and faster. Therefore, this analysis used the powerful Language R to load and transform the dataset provided, creating significant visualizations with RStudio tool. Thus, this report has the result of statistical thinking that collected meaningful data to answer the right question with detailed analysis of patterns found in the data, making possible to draw conclusions that go beyond the observed data.

Dataset description

Each event in the dataset analysed was a login to the client's app and the purpose of the analysis was to find patterns related to accounts and devices that indicate suspicious behavior, witch is possibly associated with fraud. The dataset contains records from July 2021 and it's a great dataset for evaluating linear regression models.

Initial dataset schema

- event_id: identifier of the event.
- event_timestamp: event datetime in milliseconds. **(converted to datetime)**
- account_id: identifier of the account associated with the event.
- device: identifier of the device that performed the operation.
- distance_to_frequent_location: distance (in meters) from the device, at the time of the event, to one of the frequent locations related to the account. **(converted to kilometers)**
- device_age_days: days since an account appeared related to a device.
- is_emulator: indicates whether the device is an emulator.
- has_fake_location: indicates whether the device was using false locations at the time of the operation.
- has_root_permissions: indicates whether the device has device administrator permissions.
- app_is_from_official_store: indicates whether or not the app used to perform the operation came from an official store.

Variables added to dataset schema

- event_hour: hour of the event day.
- event_date: event date (yyyy-mm-dd)
- device_age_category: variable device_age_days sliced in a category (“Day”,“Week”,“Month”,“Year”,“Year+”)
- dtfl_category: variable distance_to_frequent_location sliced in a category (“No Location”,“Frequent Location”,“Near FL”,“Far FL”)
- score_risk: variable that accumulates points whenever another variable indicates some risk
- risk_level: variable score_risk sliced in a category (“Low”,“Medium”,“High”)

Dataset processing

Loading dataset

```
tb <- read.table("Dados/hugo_incognia_db_for_da_test.csv",
                    dec = ".",
                    sep = ",",
                    h = T,
                    fileEncoding = "windows-1252")
```

Dataset general information

Dataset size is *444758* rows and *10* columns

Compactly displaying the structure of dataset

```
## 'data.frame':    444758 obs. of  10 variables:
##  $ event_id                      : chr  "59bcca18-c726" "64c883df-2a7b" "527f8468-6a6a" "e062b8d9-6b5
##  $ event_timestamp               : num  1.63e+12 1.63e+12 1.63e+12 1.63e+12 1.63e+12 ...
##  $ account_id                    : int  1893155083 2050114867 422735906 1766209306 1860027669 2562780
##  $ device                        : int  1801928094 1971517057 1956878295 1208828324 1768952722 173632
##  $ distance_to_frequent_location: num  1.85 36.26 226.41 0.11 1.73 ...
##  $ device_age_days               : int  527 184 92 366 278 524 265 181 1 505 ...
##  $ is_emulator                   : chr  "false" "false" "false" "false" ...
##  $ has_fake_location             : chr  "false" "false" "false" "false" ...
##  $ has_root_permissions          : chr  "false" "false" "false" "false" ...
##  $ app_is_from_official_store    : chr  "true" "true" "true" "true" ...
```

Dataset transformation

Removing id column, not necessary for analysis

```
tb$event_id <- NULL
```

converting variable to integer

```
tb$distance_to_frequent_location <-
  as.integer(tb$distance_to_frequent_location)
```

Converting miliseconds to timestamp

```
tb$event_timestamp <- as.POSIXct(tb$event_timestamp / 1000,
                                 origin = "1970-01-01",
                                 tz = "UTC")
summary(tb$event_timestamp)
```

```
##                  Min.               1st Qu.                Median
## "2021-07-01 00:01:21" "2021-07-08 12:59:37" "2021-07-15 13:00:40"
##                  Mean               3rd Qu.                  Max.
## "2021-07-16 02:07:13" "2021-07-23 01:21:33" "2021-07-31 23:59:58"
```

Creating variable Event Hour

```
tb$event_hour <- format(tb$event_timestamp,"%H")
tb$event_hour <- as.numeric(tb$event_hour)
```

Creating variable Event Date

```
tb$event_date <- as.Date(tb$event_timestamp,format="%Y-%m-%d")
```

Missing values

```
#sapply(tb, function(x) sum(is.na(x)))
#Amelia::missmap(tb, main = "Missing Values")
```

There are 678 missing values. However, all missing values are in a location variable, which is crucial to determine risk because it's a login attempt without location information. Solution was replace for -1 instead removing.

```
tb <- tidyr::replace_na(tb, list(distance_to_frequent_location = -1))
```

Creating a Device Age Category

```
tb$device_age_category <- cut(tb$device_age_days,
            breaks = c(0,1,7,30,365,Inf),
            labels = c("Day","Week","Month","Year","Year+"), right = FALSE)
```

Creating a Distance to frequent location Category

```
tb$dtfl_category <- cut(tb$distance_to_frequent_location,
                            breaks = c(-1,0,1,10,Inf),
                            labels = c("No Location",
                                       "Frequent Location",
                                       "Near FL",
                                       "Far FL"), right = FALSE)
```

Creating Risk Score variable

```
tb$score_risk = 0
tb$score_risk <- tb$score_risk +
  ifelse(tb$event_hour>=0 & tb$event_hour<=5,1,0)
tb$score_risk <- tb$score_risk +
  ifelse(tb$dtfl_category=="No Location",3,0)
tb$score_risk <- tb$score_risk +
  ifelse(tb$dtfl_category=="Far FL",2,0)
tb$score_risk <- tb$score_risk +
  ifelse(tb$dtfl_category=="Near FL",1,0)
tb$score_risk <- tb$score_risk +
  ifelse(tb$device_age_category=="Day",3,0)
tb$score_risk <- tb$score_risk +
  ifelse(tb$is_emulator=="true",3,0)
tb$score_risk <- tb$score_risk +
  ifelse(tb$has_fake_location=="true",3,0)
tb$score_risk <- tb$score_risk +
  ifelse(tb$has_root_permissions=="true",3,0)
tb$score_risk <- tb$score_risk +
  ifelse(tb$app_is_from_official_store=="false",3,0)
```

Creating Risk Level (category based on score)

```
tb$risk_level <- cut(tb$score_risk,
                             breaks = c(0,1,3,Inf),
                             labels = c("Low","Medium","High"), right = FALSE)
```

Creating a single column with all numeric variables and building a numeric only dataframe

```
tb_pivot <- tidyr::pivot_longer(dplyr::select
                         (dplyr::select_if
                           (tb,is.numeric),-c("account_id", "device")),
                            cols = everything(), names_to = "numeric_var",
                            values_to = "values")

tb_num <- dplyr::select(dplyr::select_if
                        (tb,is.numeric),-c("account_id", "device"))
```

Absolute and relative frequency

```
# Absolute frequency
freq1 <- table(tb[c('is_emulator')])
freq2 <- table(tb[c('has_fake_location')])
freq3 <- table(tb[c('has_root_permissions')])
freq4 <- table(tb[c('app_is_from_official_store')])
freq5 <- table(tb[c('dtfl_category')])
freq6 <- table(tb[c('device_age_category')])
freq7 <- table(tb[c('event_hour')])
freq8 <- table(tb[c('risk_level')])
```

```r
# Relative frequency
freq_rel1 <- round(prop.table(freq1),7)*100
freq_rel2 <- round(prop.table(freq2),7)*100
freq_rel3 <- round(prop.table(freq3),7)*100
freq_rel4 <- round(prop.table(freq4),7)*100
freq_rel5 <- round(prop.table(freq5),4)*100
freq_rel6 <- round(prop.table(freq6),4)*100
freq_rel7 <- round(prop.table(freq7),4)*100
freq_rel8 <- round(prop.table(freq8),4)*100
```

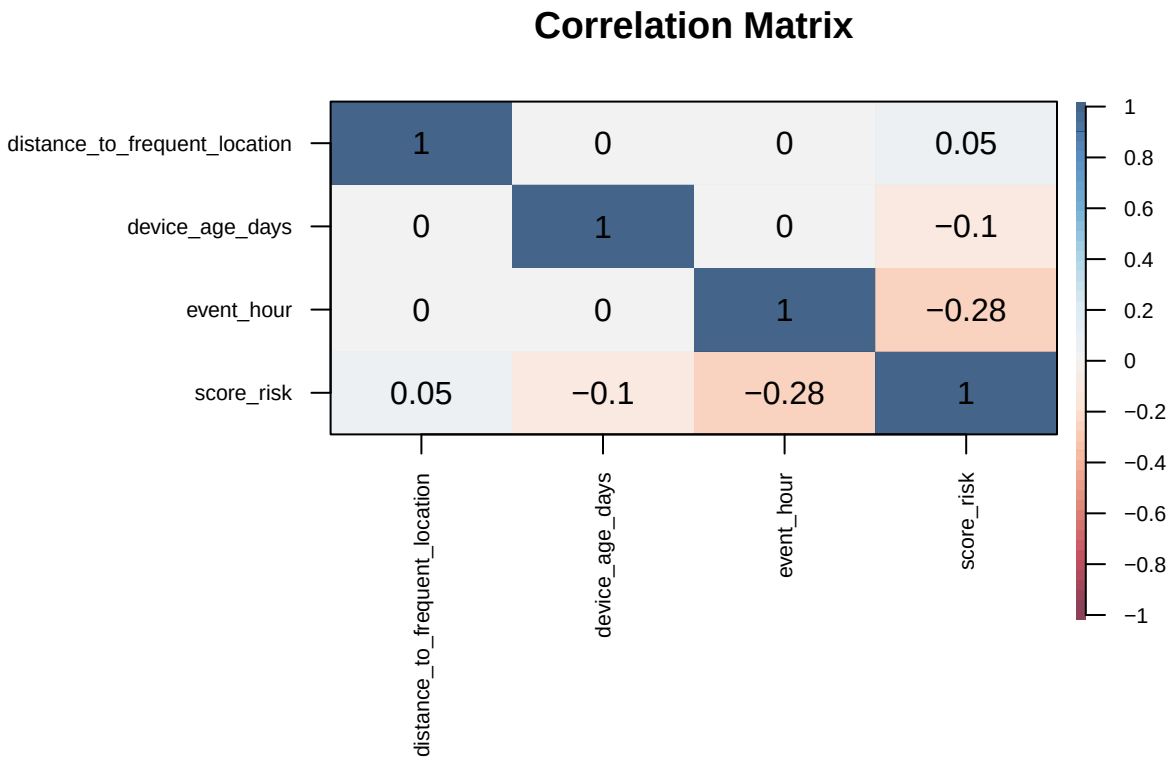
Graphical representation of information and data

Evaluating correlation between variables

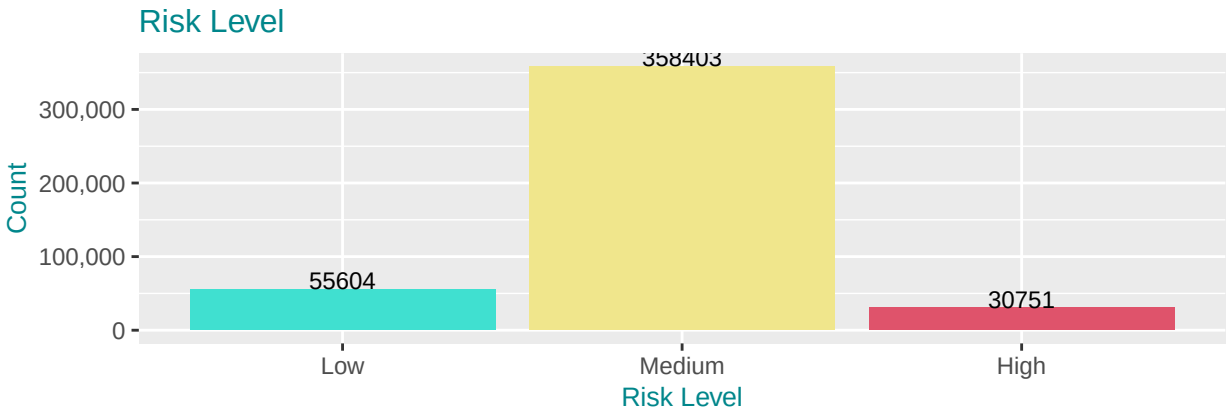
```r
#GGally::ggpairs(tb_num, upper =
#                  list(continuous =
#                         GGally::wrap("cor", size = 3))) +
#  ggplot2::theme(
#    axis.text = ggplot2::element_text(size = 6),
#    axis.title = ggplot2::element_text(size = 6),
#    legend.background = ggplot2::element_rect(fill = "white"),
#    panel.grid.major = ggplot2::element_line(colour = NA),
#    panel.grid.minor = ggplot2::element_blank(),
#    panel.background = ggplot2::element_rect(fill = "grey95"))
```

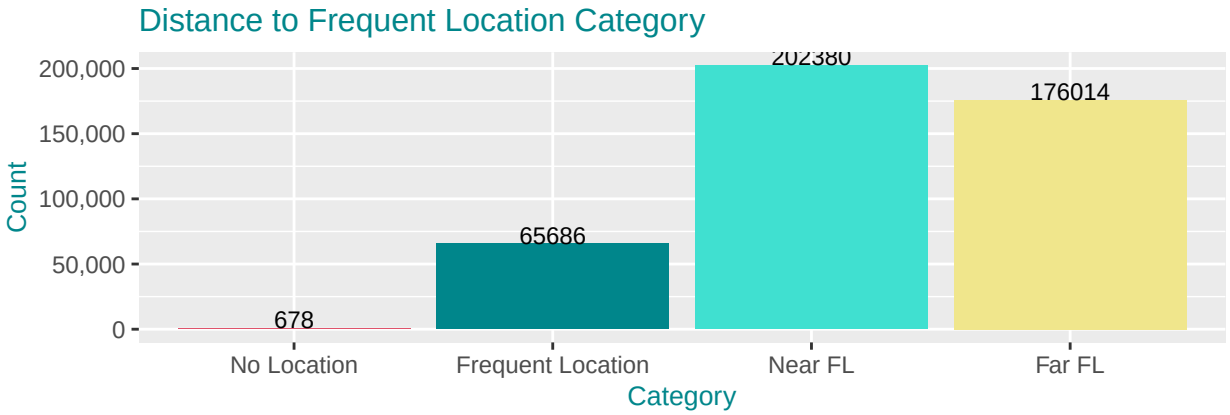
Heatmap analysis

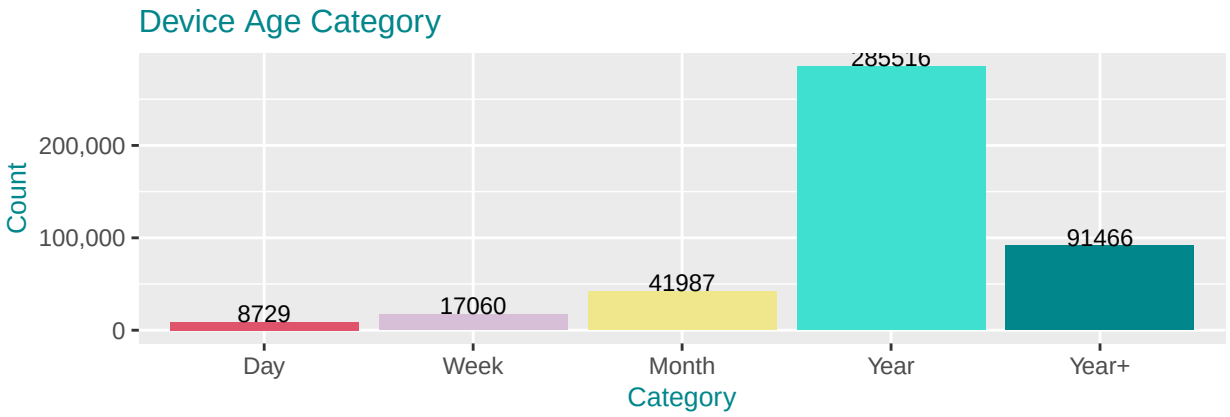
```r
psych::corPlot(tb_num, xlas=2, main="Correlation Matrix", scale=F,
        gr=colorRampPalette(c("#67001F", "#B2182B", "#D6604D", "#F4A582",
                              "#FDDBC7", "#eeeeee", "#D1E5F0", "#92C5DE",
                              "#4393C3", "#2166AC", "#053061")),
        cex=1,cex.axis=0.7)
```

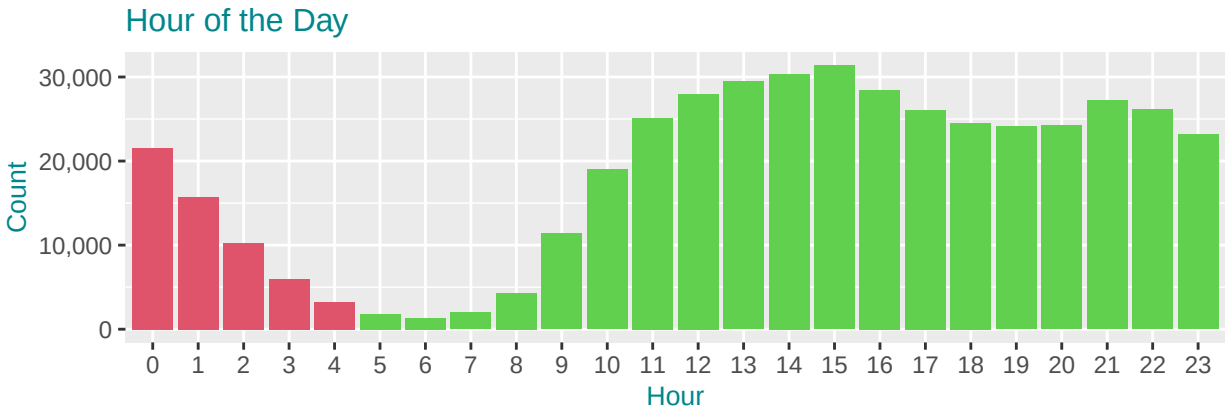


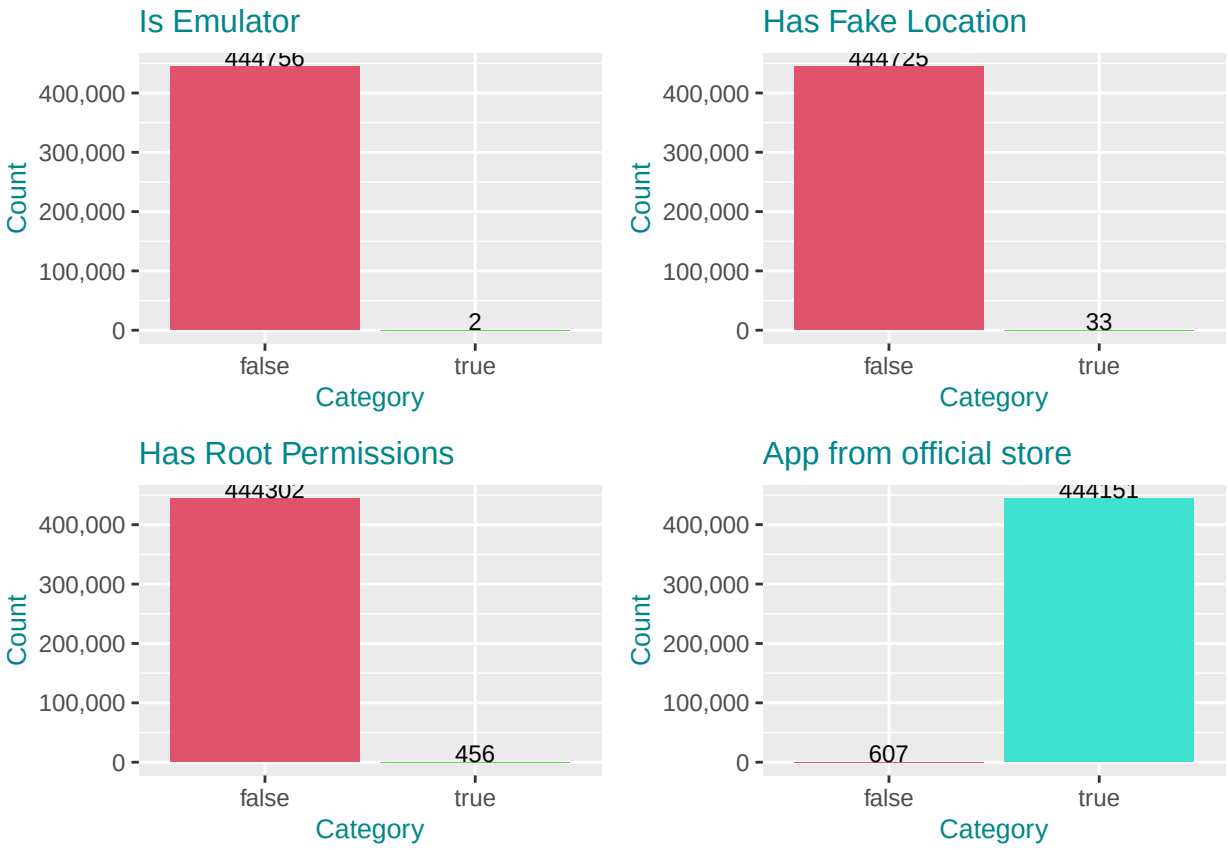
Barplot analysis


Risk Level
358403
300,000
200,000
100,000
0
Count
55604
30751
Low
Medium
High
Risk Level


Distance to Frequent Location Category
202380
200,000
150,000
100,000
50,000
0
Count
176014
65686
678
No Location
Frequent Location
Near FL
Far FL
Category

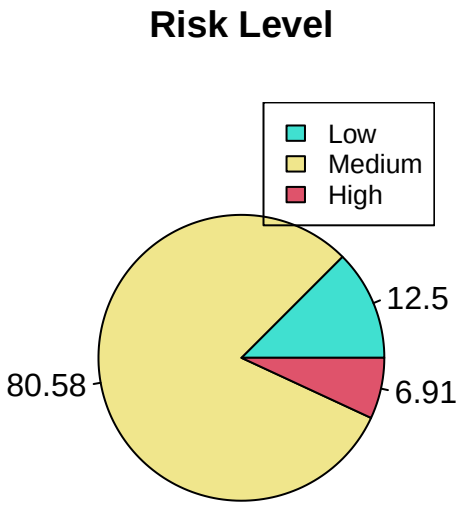

Device Age Category
285516
200,000
100,000
0
Count
8729
17060
41987
91466
Day
Week
Month
Year
Year+
Category

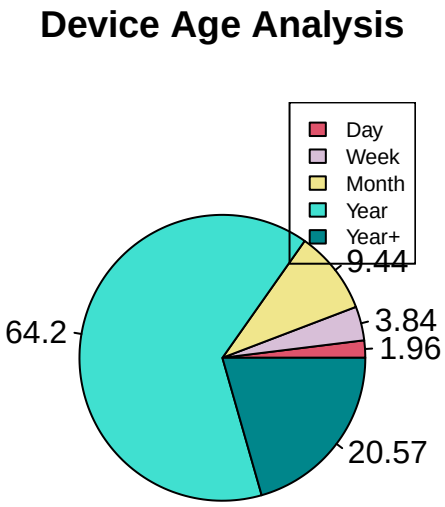

Hour of the Day
30,000
20,000
10,000
0
Count
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour



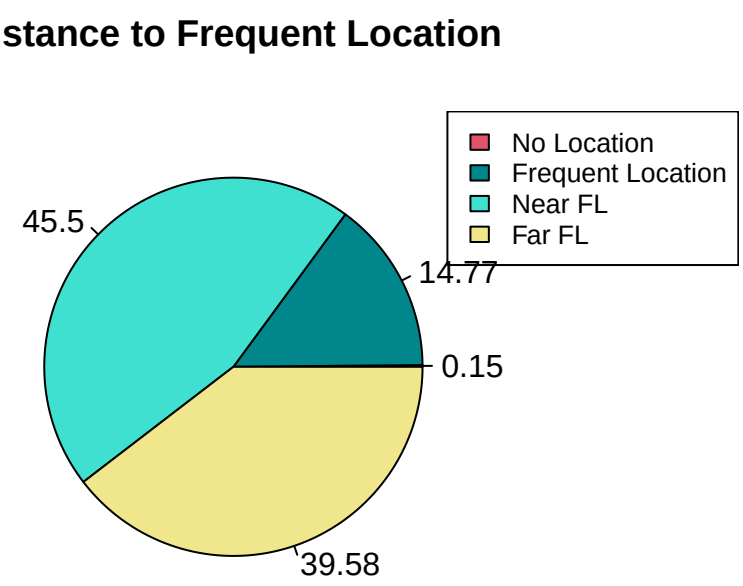
Pie Chart

```
# set the plotting area into a 1*3 array
par(mfrow=c(1,2))

pc <- pie(freq_rel8, labels = freq_rel8 ,
          main = "Risk Level",
          col = c("turquoise","khaki",2))
legend("topright", names(freq_rel8), cex = .8,
       fill = c("turquoise","khaki",2))

pc <- pie(freq_rel6, labels = freq_rel6 ,
          main = "Device Age Analysis",
          col = c(2,"thistle","khaki","turquoise","turquoise4"))
legend("topright", names(freq_rel6), cex = .7,
       fill = c(2,"thistle","khaki","turquoise","turquoise4"))
```

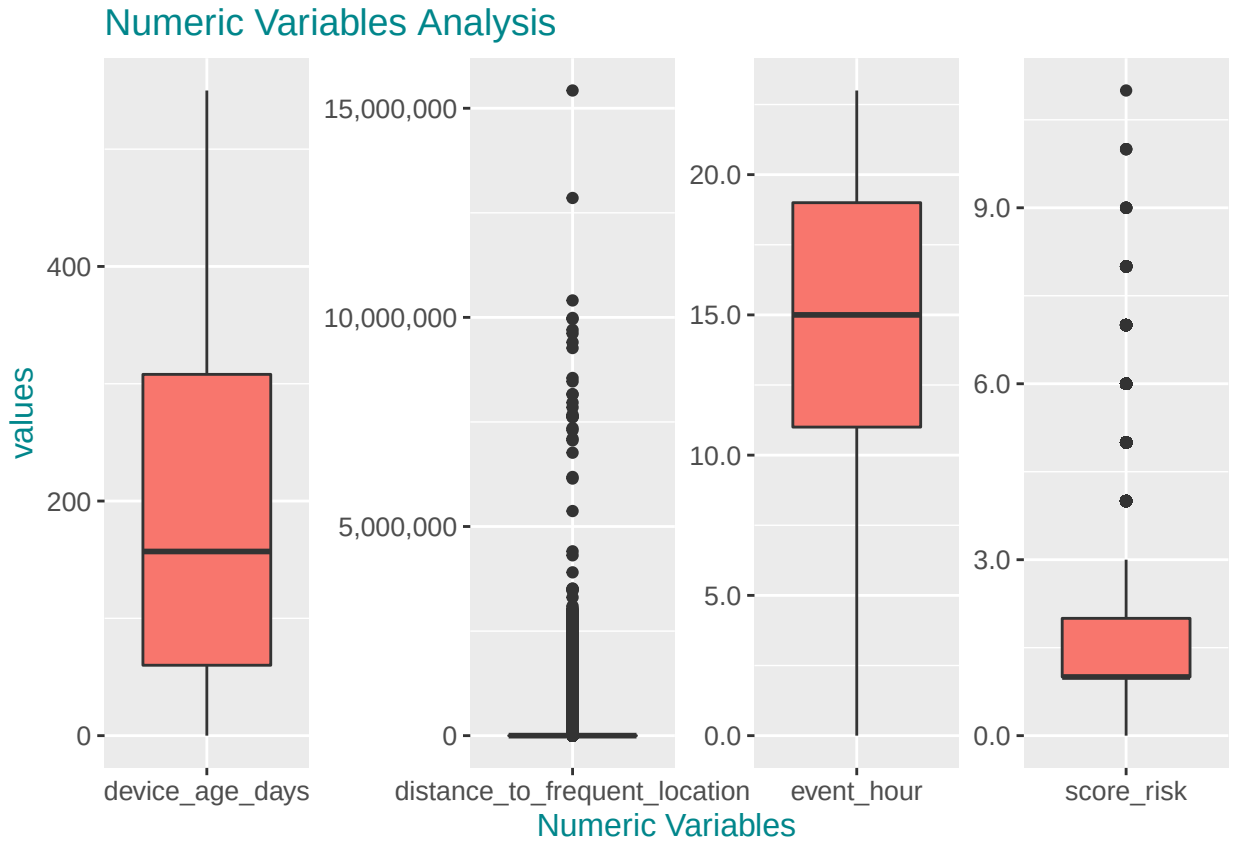


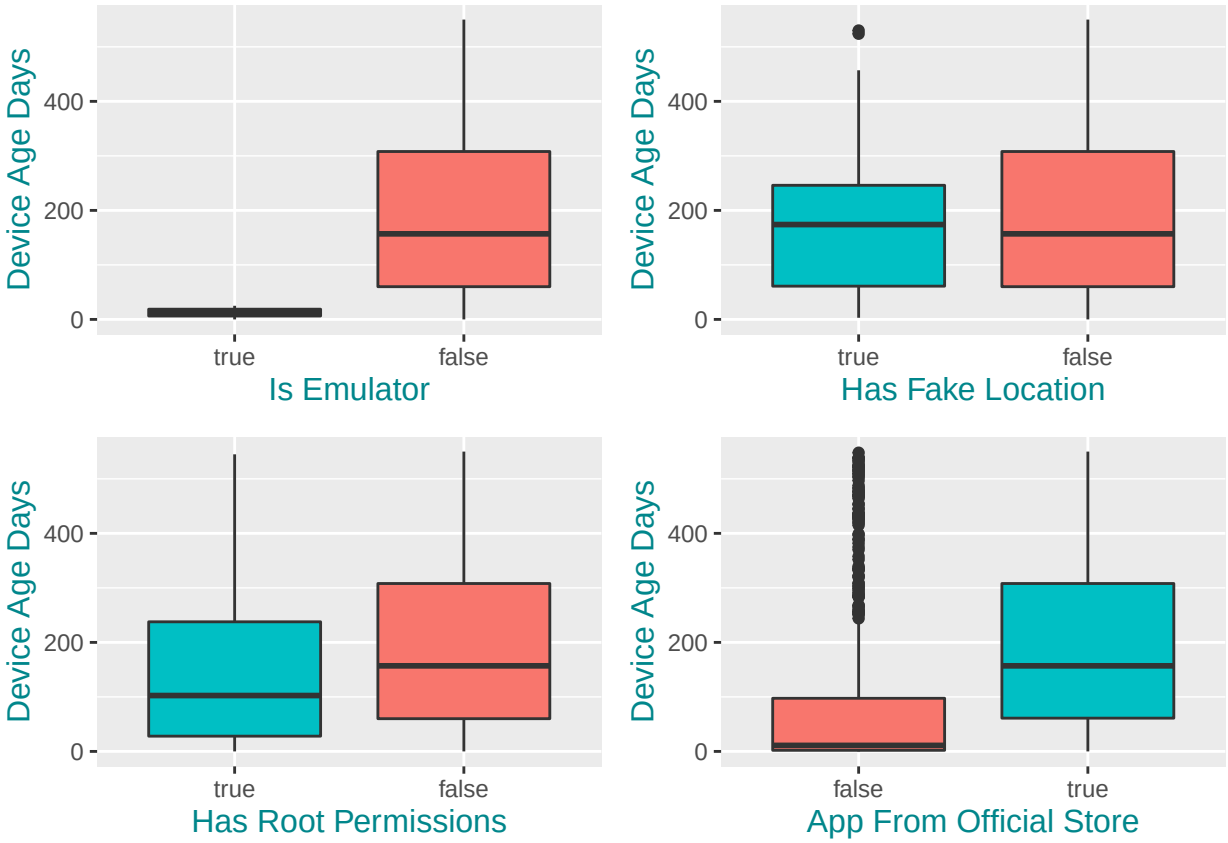


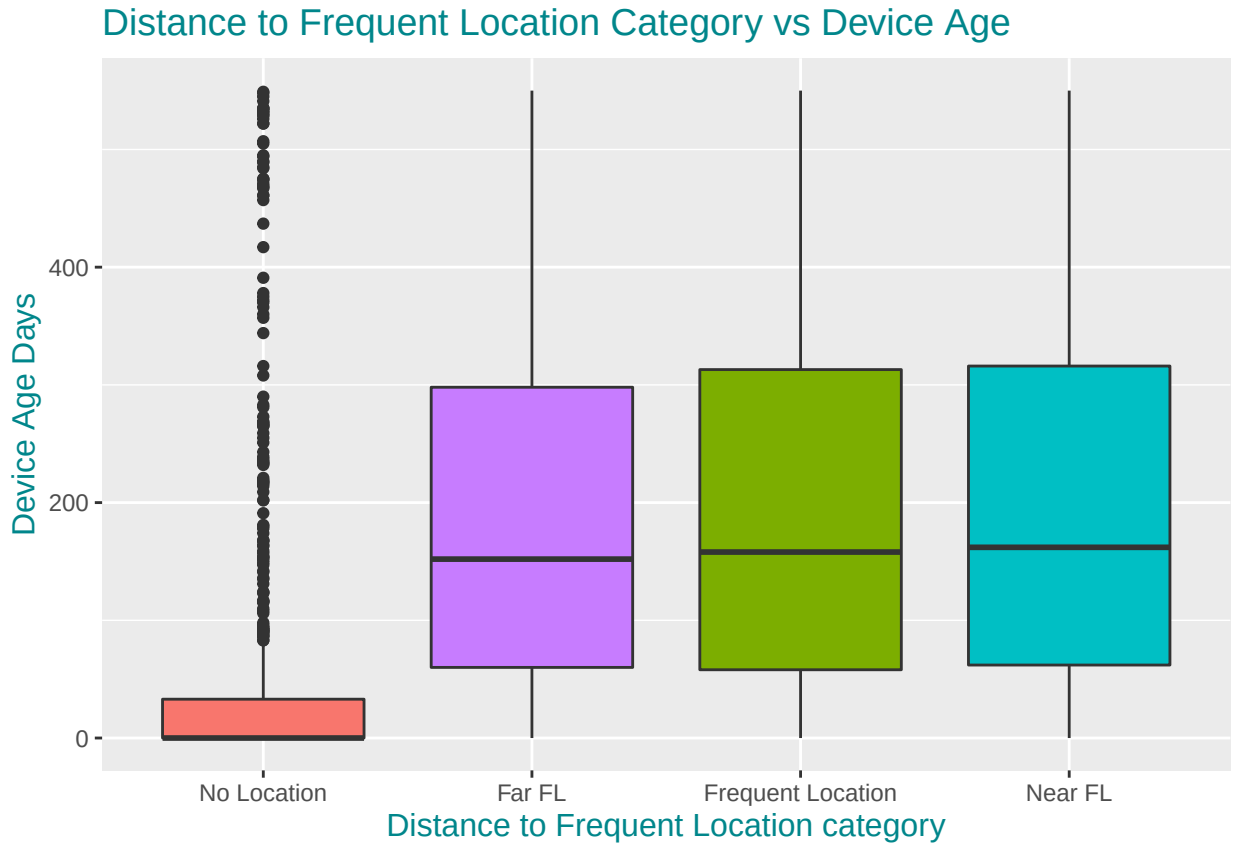

```
par(mfrow=c(1,1))
pc <- pie(freq_rel5, labels = freq_rel5 ,
          main = "Distance to Frequent Location",
          col = c(2,"turquoise4","turquoise","khaki"))
legend("topright", names(freq_rel5), cex = .8,
       fill = c(2,"turquoise4","turquoise","khaki"),)
```



Boxplot analysis


Numeric Variables Analysis
values
400
200
0
15,000,000
10,000,000
5,000,000
0
20.0
15.0
10.0
5.0
0.0
9.0
6.0
3.0
0.0
device_age_days
distance_to_frequent_location
event_hour
score_risk
Numeric Variables


Device Age Days
400
200
0
true
false
Is Emulator
Device Age Days
400
200
0
true
false
Has Fake Location
Device Age Days
400
200
0
true
false
Has Root Permissions
Device Age Days
400
200
0
false
true
App From Official Store

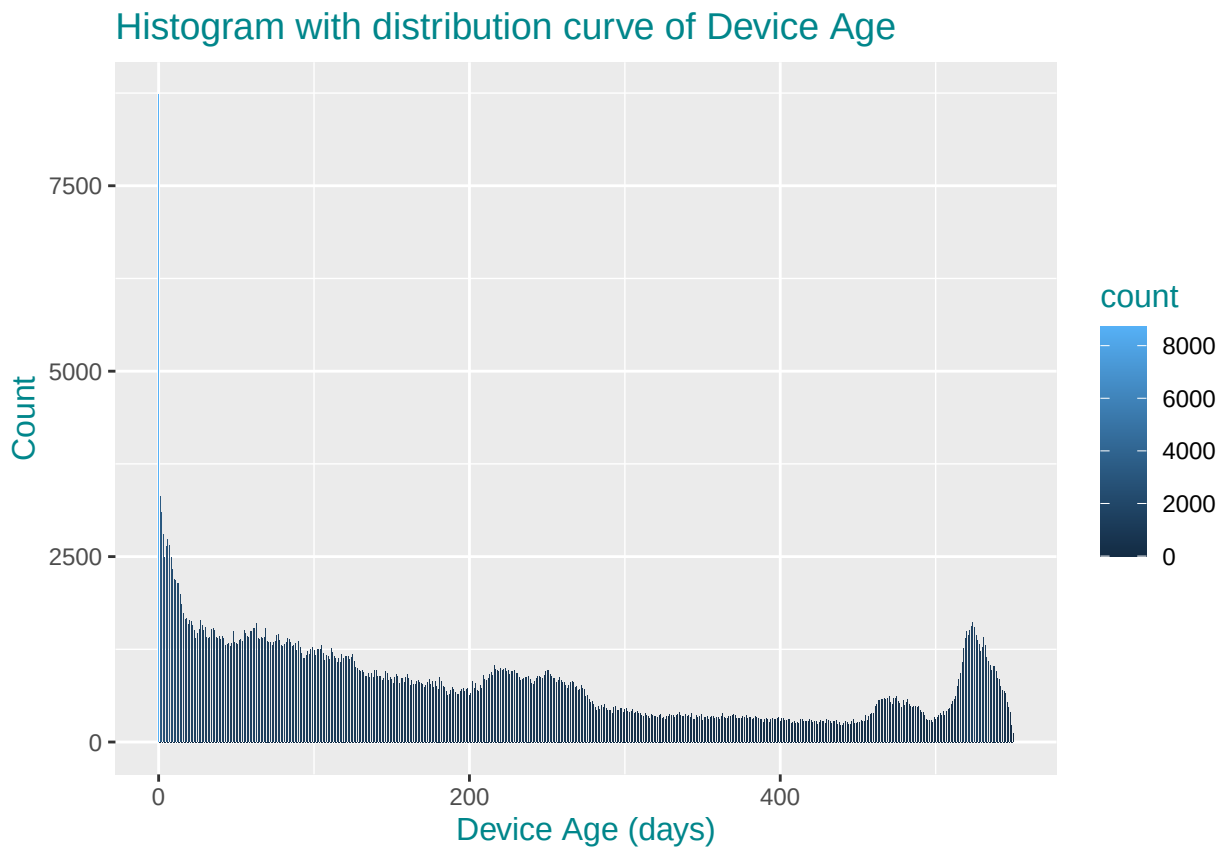


Scatter Plot analysis

```
# showing relation between two variables and a logical variable
#ggplot(data = tb, aes(y = distance_to_frequent_location,
#                       x = device_age_days)) +
#  geom_point(aes(color = risk_level), size = 2) +
#  ggtitle("Distance to frequent location vs Device Age") +
#  labs(y = "Distance to frequent location (Km)",
#       x = "Device Age (days)") +
#  theme(title = element_text(size = 12, color = "turquoise4"))
```

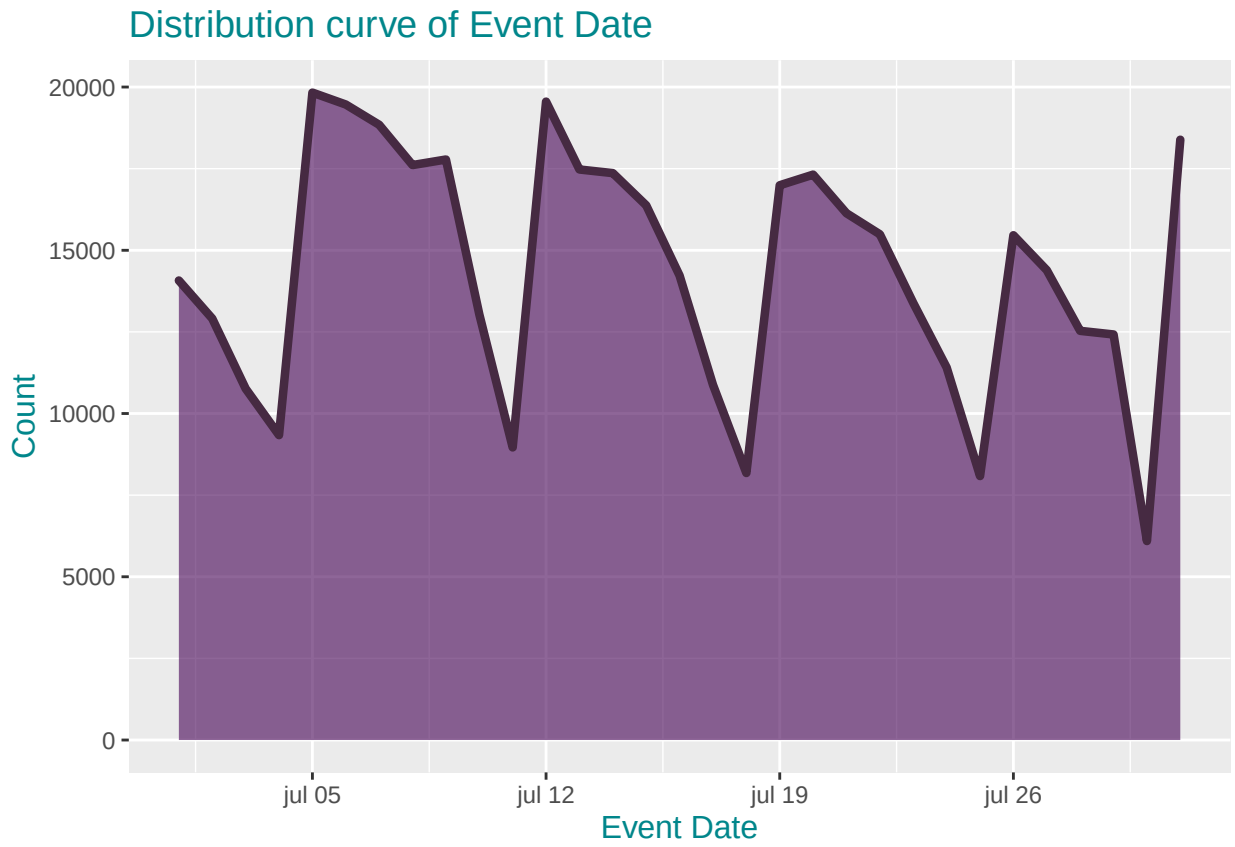
Histogram of distribution

```
# Device Age
df = data.frame(value = tb$device_age_days)
ggplot(df, aes(x=value)) +
  geom_histogram(binwidth = 0.5, aes(fill = ..count..)) +
  ggtitle("Histogram with distribution curve of Device Age") +
  labs(y = "Count",
       x = "Device Age (days)") +
  theme(title = element_text(size = 12, color = "turquoise4"))
```



Area Chart analysis

```
# Event Date
temp <- dplyr::summarise(dplyr::group_by(tb,event_date),qtde = length(event_date))
df = data.frame(value = temp)
ggplot(df, aes(x=value.event_date,y=value.qtde, fill="#462a42")) +
  geom_area(alpha=0.6 , size=1.5, colour="#462a42") +
  viridis::scale_fill_viridis(discrete = T) +
  ggtitle("Distribution curve of Event Date") +
  labs(y = "Count",
       x = "Event Date") +
  theme(legend.position='none',
        title = element_text(size = 12, color = "turquoise4"))
```



Analysis Coclusion

The dataset analysis found patterns related to the accounts and devices that indicate suspicious behavior, possibly associated with fraud. In this report, it is possible to understand how these patterns occur and how often. Moreover, with the patterns found, it was possible to create a variable that measures risk with a score points. The higher the score, the greater the risk. In addition, another variable was defined to categorize the risk level into **Low**, **Medium** and **High**. Therefore, Incognia will be able to improve the detection algorithm and with that, increase the efficiency in the communication with the financial client in order to avoid fraud.

Machine Learning Model

Objective

Create a machine learning model to predict whether a financial event is a fraud event based on patterns found in a dataset.

Preparing data to the model

Sample random rows in dataframe

Only 3000 rows were selected at random from the dataset. More than this value requires a higher computational level.

```
library(dplyr)
```

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

## The following object is masked from 'package:gridExtra':
##
##     combine

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

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

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
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(caret)
```

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

```
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:data.table':
##
##     melt

## The following object is masked from 'package:dplyr':
##
##     rename
```

```
library(randomForest)
```

```
## randomForest 4.7-1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following objects are masked from 'package:reshape':
##
##     expand, smiths
```

```
library(DMwR)
```

```
## Carregando pacotes exigidos: grid

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(stringr)
library(gridExtra)
library(scales)
library(viridis)
```

```
## Carregando pacotes exigidos: viridisLite
```

```
##
## Attaching package: 'viridis'
```

```
## The following object is masked from 'package:scales':
##
##     viridis_pal
```

```
library(hrbrthemes)
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.
```

```
##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and
```

```
##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:DMwR':
##
##     crossValidation
```

```
## The following object is masked from 'package:randomForest':
##
##     outlier
```

```
## The following objects are masked from 'package:scales':
##
##     alpha, rescale
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
# Sample random rows in dataframe
df = data.frame(tb)
tb_ml <- df[sample(nrow(df), 3000), ]

# Suggesting a variable as a possible fraud
# Risk Score greater than 3
tb_ml$is_fraud <- ifelse(tb_ml$score_risk>=4,1,0)

tb_ml$is_fraud <- as.factor(tb_ml$is_fraud)
tb_ml$account_id <- NULL #as.factor(tb_ml$account_id)
tb_ml$device <- NULL #as.factor(tb_ml$device)
tb_ml$event_hour <- as.factor(tb_ml$event_hour)
tb_ml$score_risk <- as.factor(tb_ml$score_risk)
tb_ml$event_timestamp <- NULL #as.factor(tb_ml$event_timestamp)
tb_ml$is_emulator <- as.factor(tb_ml$is_emulator)
tb_ml$has_fake_location <- as.factor(tb_ml$has_fake_location)
tb_ml$has_root_permissions <- as.factor(tb_ml$has_root_permissions)
tb_ml$app_is_from_official_store <- as.factor(tb_ml$app_is_from_official_store)
tb_ml$event_date <- NULL #as.factor(tb_ml$event_date)
#tb_ml$distance_to_frequent_location <- as.integer(tb_ml$distance_to_frequent_location)
#tb_ml$device_age_days <- as.integer(tb_ml$device_age_days)

View(tb_ml)

# Set seed
set.seed(12345)

# Selecting rows according to variable IS_FRAUD
index <- createDataPartition(tb_ml$is_fraud, p = 0.75, list = FALSE)
dim(index)
```

```
## [1] 2251    1
```

```r
# Setting training data as a subset
data_training <- tb_ml[index,]
dim(data_training)
```

```
## [1] 2251   13
```

```r
table(data_training$is_fraud)
```

```
##
##    0    1
## 2203   48
```

```r
# Percentages between classes
prop.table(table(data_training$is_fraud))
```

```
##
##          0          1
## 0.97867614 0.02132386
```

```r
# Percentage comparison between training classes and original dataset
data_comparison <- cbind(prop.table(table(data_training$is_fraud)),
                         prop.table(table(tb_ml$is_fraud)))
colnames(data_comparison) <- c("Training", "Original")
data_comparison
```

```
##     Training Original
## 0 0.97867614    0.979
## 1 0.02132386    0.021
```

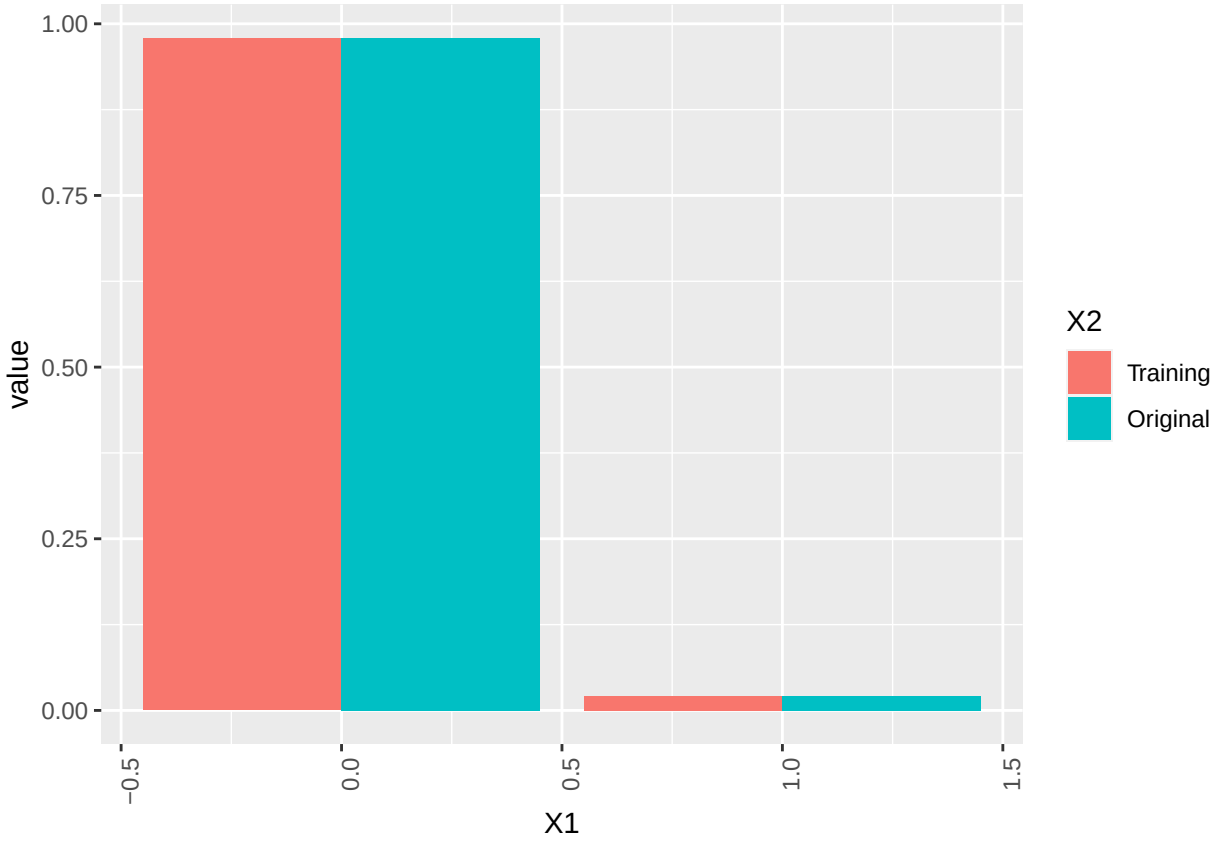
```r
# Melt Data - Convert columns to rows
melt_data_comparison <- melt(data_comparison)
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```r
melt_data_comparison
```

```
##   X1       X2      value
## 1  0 Training 0.97867614
## 2  1 Training 0.02132386
## 3  0 Original 0.97900000
## 4  1 Original 0.02100000
```

```r
# Plot - Training vs original
ggplot(melt_data_comparison, aes(x = X1, y = value)) +
  geom_bar( aes(fill = X2), stat = "identity", position = "dodge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
# Everything not in the Training Dataset must be in the Test Dataset
data_test <- tb_ml[-index,]
dim(data_test)
```

```
## [1] 749  13
```

```
dim(data_training)
```

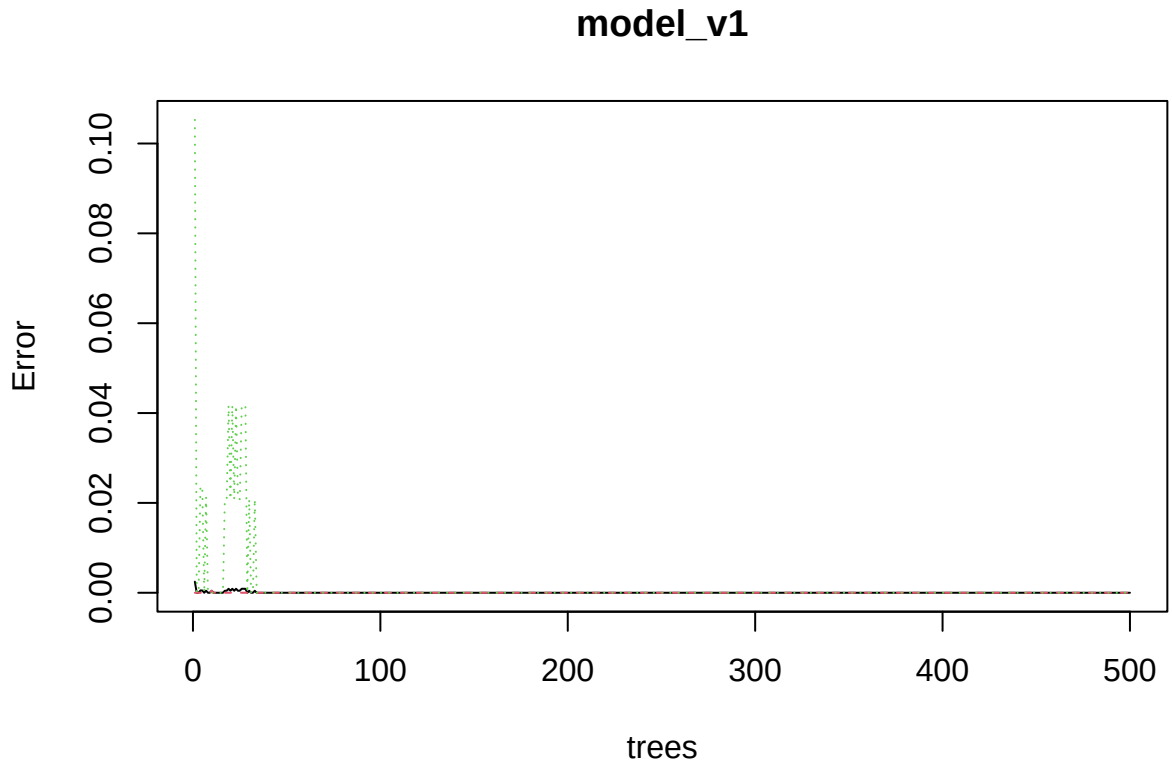
```
## [1] 2251   13
```

```
# Building model first version
View(data_training)
model_v1 <- randomForest(is_fraud ~ ., data = data_training)
model_v1
```

```
##
## Call:
##  randomForest(formula = is_fraud ~ ., data = data_training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0%
## Confusion matrix:
```

```
##      0  1 class.error
## 0 2203  0           0
## 1    0 48           0
```

```
plot(model_v1)
```




```
# Test data predicts
predict_v1 <- predict(model_v1, data_test)

# Confusion Matrix to calculate a cross-tabulation of observed and predicted classes
cm_v1 <- caret::confusionMatrix(predict_v1, data_test$is_fraud, positive = "1")
cm_v1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 734   0
##          1   0  15
##
##                Accuracy : 1
##                  95% CI : (0.9951, 1)
##     No Information Rate : 0.98
##     P-Value [Acc > NIR] : 2.627e-07
##
```

```
##                  Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.00000
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 1.00000
##              Prevalence : 0.02003
##          Detection Rate : 0.02003
##    Detection Prevalence : 0.02003
##       Balanced Accuracy : 1.00000
##
##        'Positive' Class : 1
##
```

```
# Precision, Recall e F1-Score, measures to evaluate predict model
y <- data_test$is_fraud
y_pred_v1 <- predict_v1

precision <- posPredValue(y_pred_v1, y)
precision
```

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

```
recall <- sensitivity(y_pred_v1, y)
recall
```

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

```
F1 <- (2 * precision * recall) / (precision + recall)
F1
```

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

```
# SMOTE algorithm for unbalanced classification problems

table(data_training$is_fraud)
```

```
##
##    0    1
## 2203   48
```

```
prop.table(table(data_training$is_fraud))
```

```
##
##          0          1
## 0.97867614 0.02132386
```

```r
set.seed(9560)
#str(data_test)
#temp_data_training <- data_training
#temp_data_training$app_is_from_official_store <- NULL

data_training_bal <- data_training #SMOTE(is_fraud ~ ., data  = data_training)
table(data_training_bal$is_fraud)
```

```
##
##    0    1
## 2203   48
```

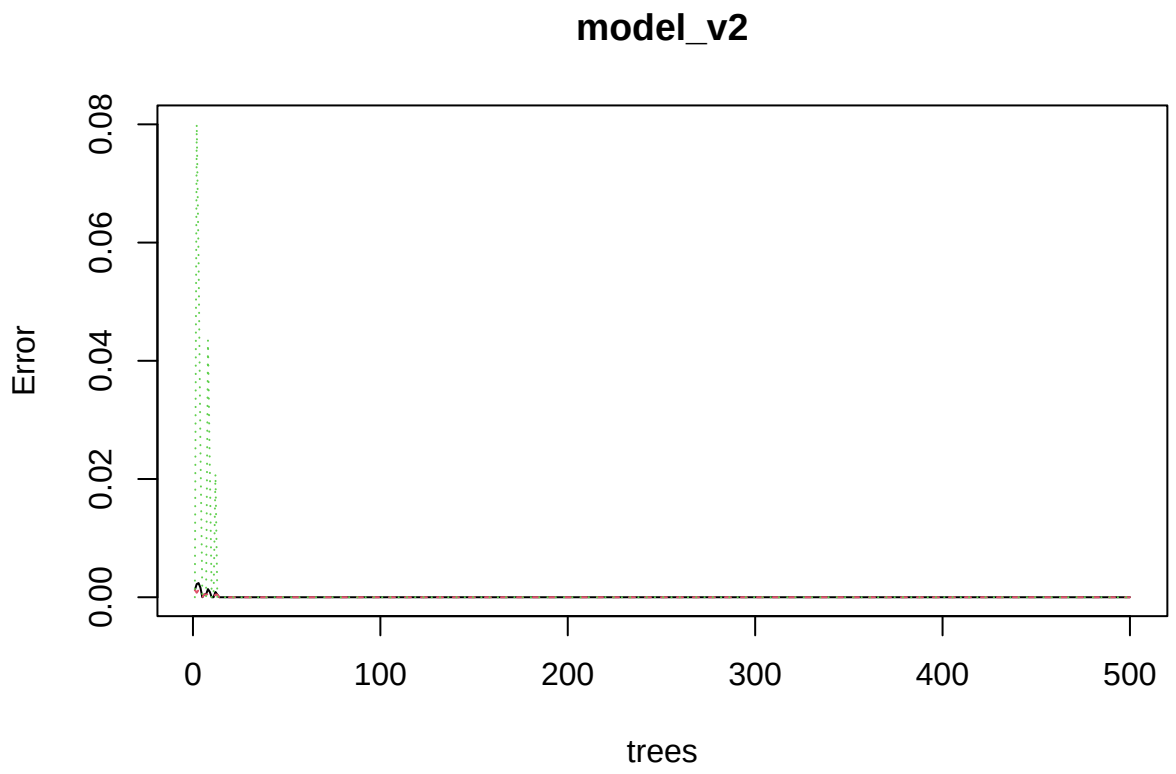
```r
prop.table(table(data_training_bal$is_fraud))
```

```
##
##          0          1
## 0.97867614 0.02132386
```

```r
# Building model version 2
model_v2 <- randomForest(is_fraud ~ ., data = data_training_bal)
model_v2
```

```
##
## Call:
##  randomForest(formula = is_fraud ~ ., data = data_training_bal)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0%
## Confusion matrix:
##      0  1 class.error
## 0 2203  0           0
## 1    0 48           0
```

```r
plot(model_v2)
```




```
# Test data predictions
predict_v2 <- predict(model_v2, data_test)

# Confusion Matrix
cm_v2 <- caret::confusionMatrix(predict_v2, data_test$is_fraud, positive = "1")
cm_v2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 734   1
##          1   0  14
##
##                Accuracy : 0.9987
##                  95% CI : (0.9926, 1)
##     No Information Rate : 0.98
##     P-Value [Acc > NIR] : 4.284e-06
##
##                   Kappa : 0.9648
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.93333
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
```

```
##          Neg Pred Value : 0.99864
##              Prevalence : 0.02003
##          Detection Rate : 0.01869
##    Detection Prevalence : 0.01869
##       Balanced Accuracy : 0.96667
##
##        'Positive' Class : 1
##
```

```
# Precision, Recall e F1-Score
y <- data_test$is_fraud
y_pred_v2 <- predict_v2

precision <- posPredValue(y_pred_v2, y)
precision
```

```
## [1] 0.9986395
```

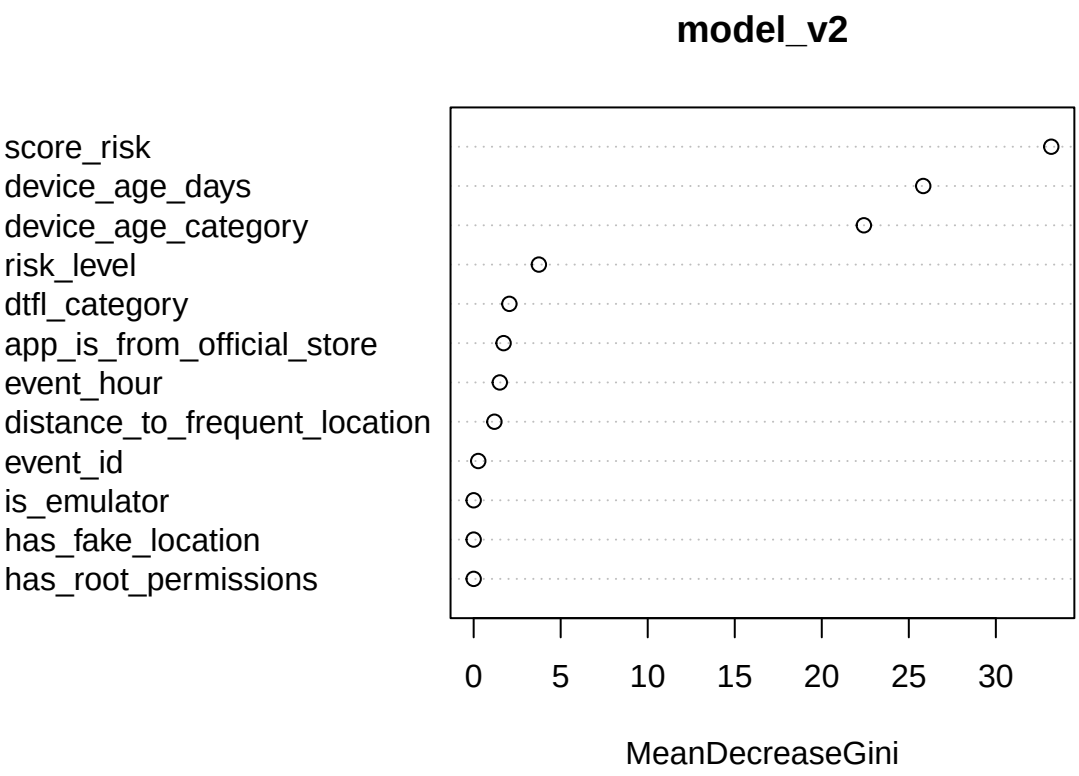
```
recall <- sensitivity(y_pred_v2, y)
recall
```

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

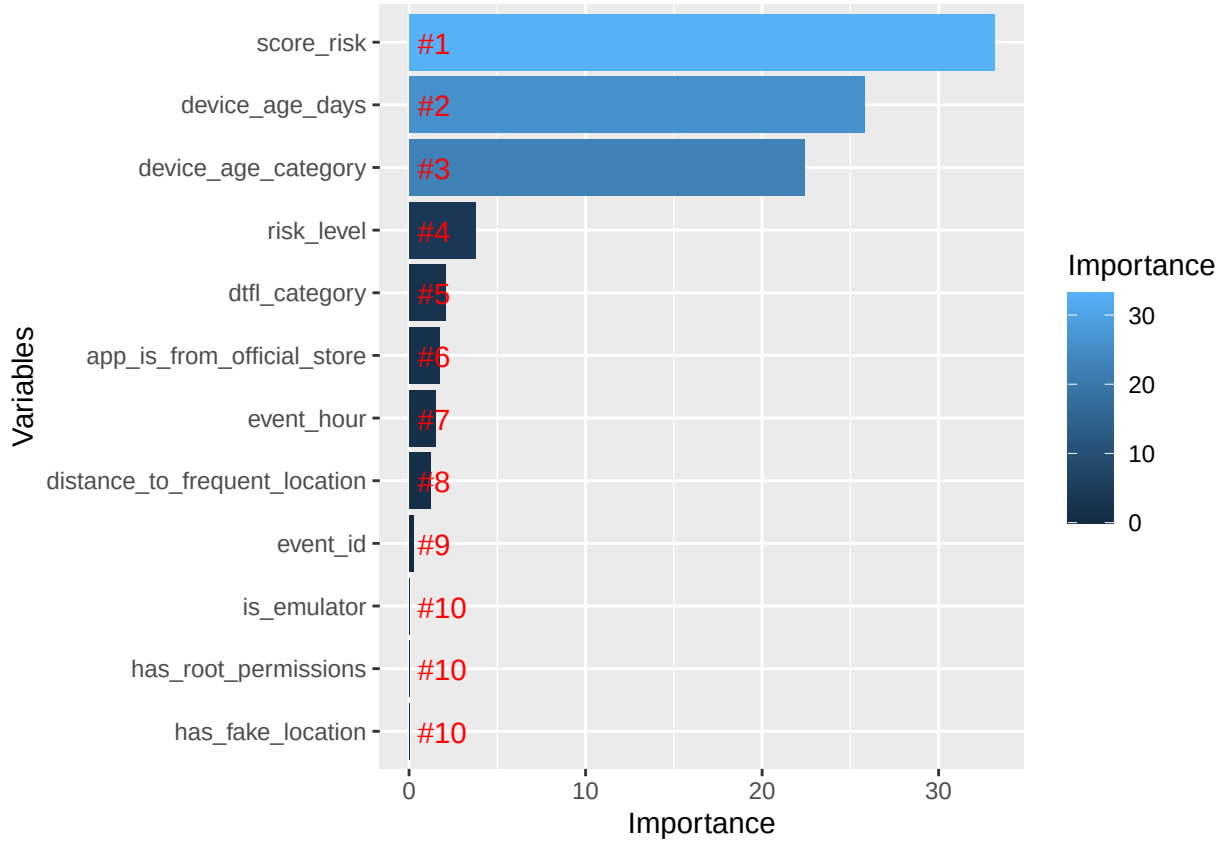
```
F1 <- (2 * precision * recall) / (precision + recall)
F1
```

```
## [1] 0.9993193
```

```
# Most important variables to predict
varImpPlot(model_v2)
```




```
imp_var <- importance(model_v2)
varImportance <- data.frame(Variables = row.names(imp_var),
                            Importance = round(imp_var[ ,'MeanDecreaseGini'],2))
rankImportance <- varImportance %>%
  mutate(Rank = paste0('#', dense_rank(desc(Importance))))

# Visualizing related importance
ggplot(rankImportance,
       aes(x = reorder(Variables, Importance),
           y = Importance,
           fill = Importance)) +
  geom_bar(stat='identity') +
  geom_text(aes(x = Variables, y = 0.5, label = Rank),
            hjust = 0,
            vjust = 0.55,
            size = 4,
            colour = 'red') +
  labs(x = 'Variables') +
  coord_flip()
```




```
# Building model version 3 with the most important variables
colnames(data_training_bal)
```

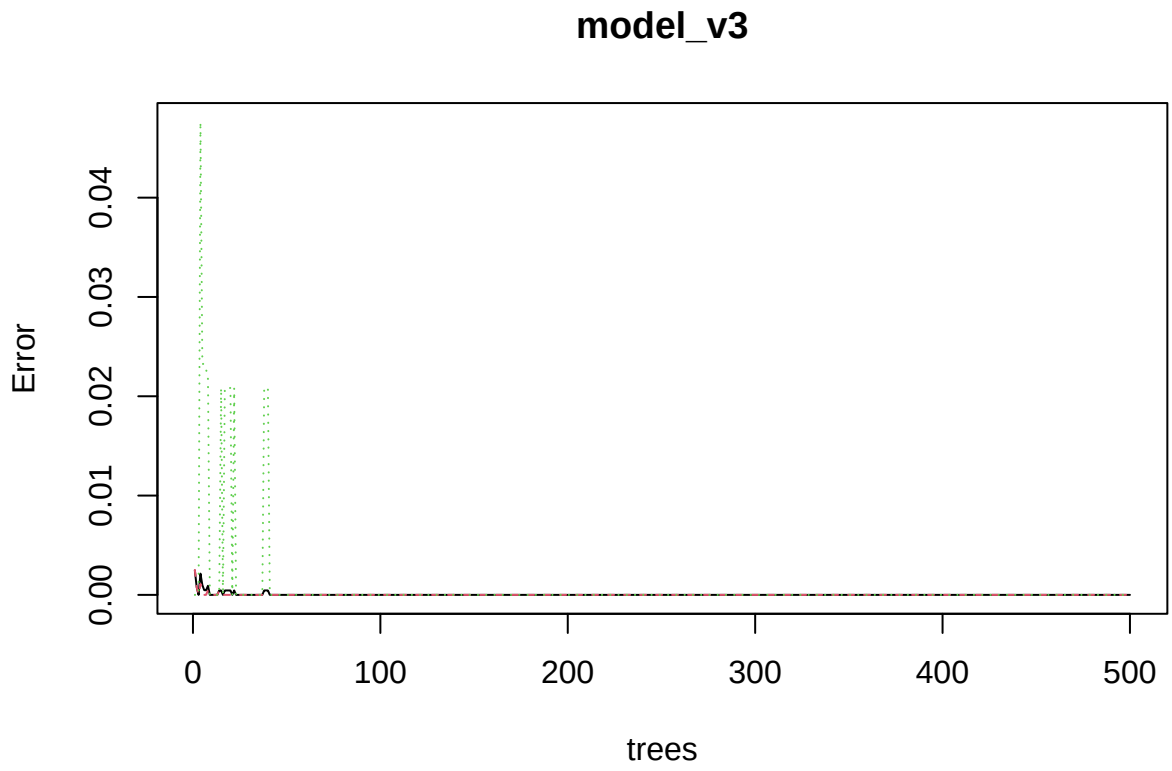
```
##  [1] "event_id"                      "distance_to_frequent_location"
##  [3] "device_age_days"               "is_emulator"
##  [5] "has_fake_location"             "has_root_permissions"
##  [7] "app_is_from_official_store"    "event_hour"
##  [9] "device_age_category"           "dtfl_category"
## [11] "score_risk"                    "risk_level"
## [13] "is_fraud"
```

```
model_v3 <- randomForest(is_fraud ~ device_age_days + device_age_category + score_risk + risk_level + ev
                         data = data_training_bal)
model_v3
```

```
##
## Call:
##  randomForest(formula = is_fraud ~ device_age_days + device_age_category +      score_risk + risk_lev
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0%
## Confusion matrix:
##      0  1 class.error
```

```
## 0 2203  0           0
## 1    0 48           0
```

```
plot(model_v3)
```




```
# Test data predictions
predict_v3 <- predict(model_v3, data_test)

# Confusion Matrix
cm_v3 <- caret::confusionMatrix(predict_v3, data_test$is_fraud, positive = "1")
cm_v3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 734   0
##          1   0  15
##
##                Accuracy : 1
##                  95% CI : (0.9951, 1)
##     No Information Rate : 0.98
##     P-Value [Acc > NIR] : 2.627e-07
##
##                   Kappa : 1
```

```
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.00000
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 1.00000
##              Prevalence : 0.02003
##          Detection Rate : 0.02003
##    Detection Prevalence : 0.02003
##       Balanced Accuracy : 1.00000
##
##        'Positive' Class : 1
##
```

```
# Precision, Recall e F1-Score
y <- data_test$is_fraud
y_pred_v3 <- predict_v3

precision <- posPredValue(y_pred_v3, y)
precision
```

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

```
recall <- sensitivity(y_pred_v3, y)
recall
```

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

```
F1 <- (2 * precision * recall) / (precision + recall)
F1
```

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

```
# Salve model file
#saveRDS(model_v3, file = "Dados/model_v3.rds")

# Loading model
#final_model <- readRDS("Dados/model_v3.rds")
final_model <- model_v3

# Predictions with new data from 3 events

# Events data
device_age_days <- c(0, 10, 100)
device_age_category <- c("Day", "Month", "Year")
score_risk <- c("3", "0", "0")
risk_level <- c("High", "Low", "Low")
event_hour <- c("4", "10", "11")
distance_to_frequent_location <- c(1000.0, 0.001, 0.001)

new_events <- data.frame(device_age_days, device_age_category, score_risk, risk_level, event_hour, dista
```

```
new_events$device_age_category <- factor(new_events$device_age_category, levels = levels(data_training_
new_events$score_risk <- factor(new_events$score_risk, levels = levels(data_training_bal$score_risk))
new_events$risk_level <- factor(new_events$risk_level, levels = levels(data_training_bal$risk_level))
new_events$event_hour <- factor(new_events$event_hour, levels = levels(data_training_bal$event_hour))

# Predictions
pred_new_events <- predict(final_model, new_events)
df = data.frame(pred_new_events)
df$is_fraud <- ifelse(pred_new_events==0,"No","Yes")
View(df)
```

Model Results

To create a machine learning model it would be important to have a sample with descriptive information determining whether each event is a fraud event. With this information, it would be possible to create a prediction algorithm model to predict which future events will be a fraud event based on the training and testing model.

As the dataset did not contain the information (yes/no fraud), then a risk level was suggested based on the patterns found. An experiment was done based on the hypothesis that a score of 4+ is a fraud event. However, the rules that defined this score make the confidence level of the experiment result in 100%. That is, the model does not make mistakes in the predictions. Therefore, the ideal is to have the information (yes/no fraud) to create a predictive model without being biased.