Universidad de Granada

Sistemas Inteligentes para la Gestión en la Empresa

Práctica 1: Pre-procesamiento de datos y clasificación binaria

Curso 2019-2020

Realizado por:

Francisco Solano López Rodríguez
20100444P fransol0728@correo.ugr.es

Máster universitario en Ingeniería Informática

Índice

1. Introducción

Esta práctica consiste en la resolución de un problema de pre-procesamiento y aprendizaje automático, en el cual trabajaremos con un conjunto de datos correspondientes a transacciones online, donde tendremos datos de la propia transacción y de la identidad de la persona que realiza la transacción.

El objetivo que se persigue es el de intentar predecir si la transacción online es fraudulenta o no, por lo que se trata de un problema de clasificación binaria. Una de las dificultades que presenta este problema es el de tener un dataset enormemente desbalanceado, ya que por lo general lo más común es que no haya fraude. Otro de las dificultades a las que nos enfrentaremos, es el de la interpretación de las características, ya que muchas de ellas están enmascaradas para proteger la privacidad de las personas.

En la página web de Kaggle podemos ver información acerca de los datos, en particular se informa de que las siguientes variables son categóricas, a pesar de que algunas de ellas solo presentan valores numéricos:

- **Variables categóricas en el fichero de transacciones**:
  ProductCD, card1 - card6, addr1, addr2, P_emaildomain, R_emaildomain, M1 - M9.
- **Variables categóricas en el fichero de identidades**:
  DeviceType, DeviceInfo, id_12 - id_38.

Los ficheros de transacciones e identidad se pueden unir por la variable TransactionDT, que en nuestro caso no será necesario, ya que se nos han proporcionado unidos.

En el siguiente link de Kaggle podemos ver información adicional acerca de los datos: descripción de los datos. En este link podemos ver el significado de algunas variables, lo cual nos va a permitir entender mejor el problema. Por ejemplo se nos revela que los datos card1 - card6 corresponden a información como el tipo de tarjeta, categoría, banco emisor o país.

2. Análisis exploratorio

En esta sección procederemos a realizar un análisis exploratorio de los datos, de forma que tengamos un mejor entendimiento de los mismos y además nos ayude a plantear la limpieza de los datos que realizaremos más tarde.

En primer lugar vamos a leer los datos del fichero csv.

```
df<-read.csv(file="./data/train_innerjoin.csv", header=TRUE, sep=",")
```

Veamos cual es la dimsión del dataset del que disponemos.

```
# Ver la dimension del dataset
dim(df)
```

```
[1] 144233    434
```

Como podemos ver tenemos un dataset con una dimensión de características bastante elevada, teniendo un total de 434 variables, de las cuales una es la de la clase a predecir. Además tenemos 144233 filas, cada una correspondiente con una transacción.

Como diponemos de muchas filas podría ser interesante ver si hay duplicados y así reducir el tamaño del dataset. Para ello borramos la variable TransactionID y vemos si hay filas duplicadas.

```
df <- subset(df, select = TransactionID)
sum(duplicated(df))
```

```
[1] 0
```

Como podemos ver el resultado es 0, por lo que no tenemos ninguna fila duplicada en el conjunto de datos.

Veamos ahora si tenemos valores NA.

```
df_status(df)
```

```
          variable q_zeros p_zeros   q_na   p_na q_inf p_inf    type unique
1          isFraud  132915   92.15      0   0.00     0     0 integer      2
2    TransactionDT       0    0.00      0   0.00     0     0 integer 142734
3   TransactionAmt       0    0.00      0   0.00     0     0 numeric   7830
4        ProductCD       0    0.00      0   0.00     0     0  factor      4
5            card1       0    0.00      0   0.00     0     0 integer   8499
6            card2       0    0.00    902   0.63     0     0 integer    482
7            card3       0    0.00    172   0.12     0     0 integer    103
8            card4       0    0.00    184   0.13     0     0  factor      4
9            card5       0    0.00    956   0.66     0     0 integer    106
```

10	card6	0	0.00	178	0.12	0	0	factor	3
11	addr1	0	0.00	60447	41.91	0	0	integer	255
12	addr2	0	0.00	60447	41.91	0	0	integer	72
13	dist1	0	0.00	144233	100.00	0	0	logical	0
14	dist2	3516	2.44	106640	73.94	0	0	integer	1751
15	P_emaildomain	0	0.00	13391	9.28	0	0	factor	59
16	R_emaildomain	0	0.00	13150	9.12	0	0	factor	60
17	C1	447	0.31	0	0.00	0	0	integer	1559
18	C2	267	0.19	0	0.00	0	0	integer	1182
19	C3	141815	98.32	0	0.00	0	0	integer	27
20	C4	11972	8.30	0	0.00	0	0	integer	1202
21	C5	144233	100.00	0	0.00	0	0	integer	1
22	C6	12251	8.49	0	0.00	0	0	integer	1194
23	C7	82967	57.52	0	0.00	0	0	integer	1036
24	C8	3328	2.31	0	0.00	0	0	integer	1236
25	C9	144233	100.00	0	0.00	0	0	integer	1
26	C10	8991	6.23	0	0.00	0	0	integer	1211
27	C11	1020	0.71	0	0.00	0	0	integer	1354
28	C12	82973	57.53	0	0.00	0	0	integer	1126
29	C13	20074	13.92	0	0.00	0	0	integer	1362
30	C14	20547	14.25	0	0.00	0	0	integer	994
31	D1	114399	79.32	218	0.15	0	0	integer	641
32	D2	6078	4.21	113117	78.43	0	0	integer	641
33	D3	14544	10.08	115174	79.85	0	0	integer	543
34	D4	42386	29.39	79465	55.09	0	0	integer	791
35	D5	18194	12.61	111158	77.07	0	0	integer	619
36	D6	41796	28.98	76860	53.29	0	0	integer	828
37	D7	19675	13.64	108093	74.94	0	0	integer	595
38	D8	835	0.58	69307	48.05	0	0	numeric	12353
39	D9	4190	2.91	69307	48.05	0	0	numeric	24
40	D10	63232	43.84	75001	52.00	0	0	integer	781
41	D11	0	0.00	144233	100.00	0	0	logical	0
42	D12	38779	26.89	85324	59.16	0	0	integer	632
43	D13	48829	33.85	82297	57.06	0	0	integer	577
44	D14	45251	31.37	82068	56.90	0	0	integer	802
45	D15	47836	33.17	75916	52.63	0	0	integer	849
46	M1	0	0.00	144233	100.00	0	0	logical	0
47	M2	0	0.00	144233	100.00	0	0	logical	0
48	M3	0	0.00	144233	100.00	0	0	logical	0
49	M4	0	0.00	83402	57.82	0	0	factor	3
50	M5	0	0.00	144233	100.00	0	0	logical	0
51	M6	0	0.00	144233	100.00	0	0	logical	0
52	M7	0	0.00	144233	100.00	0	0	logical	0
53	M8	0	0.00	144233	100.00	0	0	logical	0
54	M9	0	0.00	144233	100.00	0	0	logical	0
55	V1	0	0.00	144233	100.00	0	0	logical	0
56	V2	0	0.00	144233	100.00	0	0	logical	0
57	V3	0	0.00	144233	100.00	0	0	logical	0
58	V4	0	0.00	144233	100.00	0	0	logical	0
59	V5	0	0.00	144233	100.00	0	0	logical	0
60	V6	0	0.00	144233	100.00	0	0	logical	0
61	V7	0	0.00	144233	100.00	0	0	logical	0
62	V8	0	0.00	144233	100.00	0	0	logical	0
63	V9	0	0.00	144233	100.00	0	0	logical	0
64	V10	0	0.00	144233	100.00	0	0	logical	0
65	V11	0	0.00	144233	100.00	0	0	logical	0
66	V12	69181	47.96	75052	52.04	0	0	integer	1
67	V13	69181	47.96	75052	52.04	0	0	integer	1
68	V14	254	0.18	75052	52.04	0	0	integer	2
69	V15	12360	8.57	75052	52.04	0	0	integer	8
70	V16	12358	8.57	75052	52.04	0	0	integer	15

```
71              V17    2689    1.86  75052  52.04     0     0 integer     16
72              V18    2660    1.84  75052  52.04     0     0 integer     16
73              V19    6670    4.62  75052  52.04     0     0 integer      8
74              V20    6668    4.62  75052  52.04     0     0 integer     15
75              V21    3084    2.14  75052  52.04     0     0 integer      6
76              V22    3051    2.12  75052  52.04     0     0 integer      9
77              V23     342    0.24  75052  52.04     0     0 integer     14
78              V24     333    0.23  75052  52.04     0     0 integer     14
79              V25    2769    1.92  75052  52.04     0     0 integer      6
80              V26    2767    1.92  75052  52.04     0     0 integer     11
81              V27   68792   47.70  75052  52.04     0     0 integer      4
82              V28   68789   47.69  75052  52.04     0     0 integer      4
83              V29   69181   47.96  75052  52.04     0     0 integer      1
84              V30   69181   47.96  75052  52.04     0     0 integer      1
85              V31    3358    2.33  75052  52.04     0     0 integer      8
86              V32    3352    2.32  75052  52.04     0     0 integer     15
87              V33   12362    8.57  75052  52.04     0     0 integer      6
88              V34   12360    8.57  75052  52.04     0     0 integer     11
89              V35   64745   44.89  79488  55.11     0     0 integer      1
90              V36   64745   44.89  79488  55.11     0     0 integer      1
91              V37     303    0.21  79488  55.11     0     0 integer     55
92              V38     303    0.21  79488  55.11     0     0 integer     55
93              V39    2187    1.52  79488  55.11     0     0 integer     16
94              V40    2046    1.42  79488  55.11     0     0 integer     18
95              V41     291    0.20  79488  55.11     0     0 integer      2
96              V42    2616    1.81  79488  55.11     0     0 integer      9
97              V43    2454    1.70  79488  55.11     0     0 integer      9
98              V44     344    0.24  79488  55.11     0     0 integer     49
99              V45     326    0.23  79488  55.11     0     0 integer     49
100             V46     330    0.23  79488  55.11     0     0 integer      7
101             V47     329    0.23  79488  55.11     0     0 integer      9
102             V48   64745   44.89  79488  55.11     0     0 integer      1
103             V49   64745   44.89  79488  55.11     0     0 integer      1
104             V50    1401    0.97  79488  55.11     0     0 integer      6
105             V51    6183    4.29  79488  55.11     0     0 integer      7
106             V52    6183    4.29  79488  55.11     0     0 integer      9
107             V53   67706   46.94  76527  53.06     0     0 integer      1
108             V54   67706   46.94  76527  53.06     0     0 integer      1
109             V55     173    0.12  76527  53.06     0     0 integer     18
110             V56     173    0.12  76527  53.06     0     0 integer     52
111             V57    9778    6.78  76527  53.06     0     0 integer      7
 [ reached 'max' / getOption("max.print") -- omitted 322 rows ]
```

No se han mostrado todas las variables, pero con las que se muestran aquí podemos ver que la respuesta es afirmativa respecto a la pregunta de si hay valores perdidos, lo cual se puede consultar en las columnas **q_na** y **p_na** que nos muestran la cifra total y el porcentaje de valores na en cada variable. De hecho podemos ver que incluso hay atributos con un 100 % de valores perdidos, como por ejemplo el atributo M1. Las columnas con un porcentaje muy elevado de valores na serán eliminadas en el preprocesamiento.

La función **df_status** también nos muestra el porcentaje de ceros que hay en una columna, como podemos ver algunas variables tienen un valor muy cercano al 100 % y serán también buenas candidatas para ser eliminadas. Asimismo podemos ver en la columna **unique**, la cantidad de valores distintos que presenta una variable. Como podemos observar, algunos atributos presentan un único valor, es decir, se repite siempre el mismo valor en cada fila, lo cual no aporta nada de información y por tanto serán variables que deberán ser eliminadas también.

Además en la columna **type**, podemos observar que algunas de las variables que se describían como categóricas en la descipción de Kaggle, aparecen como numéricas, por lo tanto deberemos transformarlas en factores en el preprocesamiento.

Analicemos ahora la proporción de transacciones fraudulentas frente a las legítimas.

```
library(ggplot2)

# Convertir isFraud a factor
df <- df %>% mutate(isFraud = as.factor(ifelse(isFraud == 1, 'Yes', '
    No')))

# Propocion de fraude
summary(df$isFraud)
ggplot(df, aes(x=isFraud, color=isFraud, fill=isFraud)) +
  geom_bar()
```

```
    No    Yes
132915  11318
```

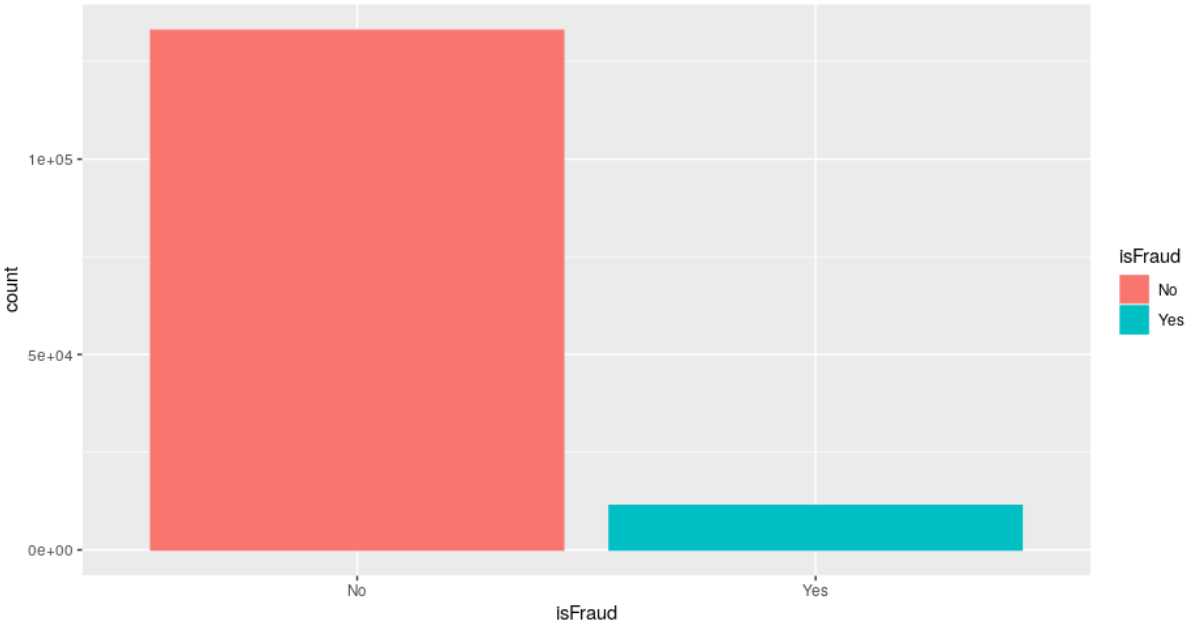


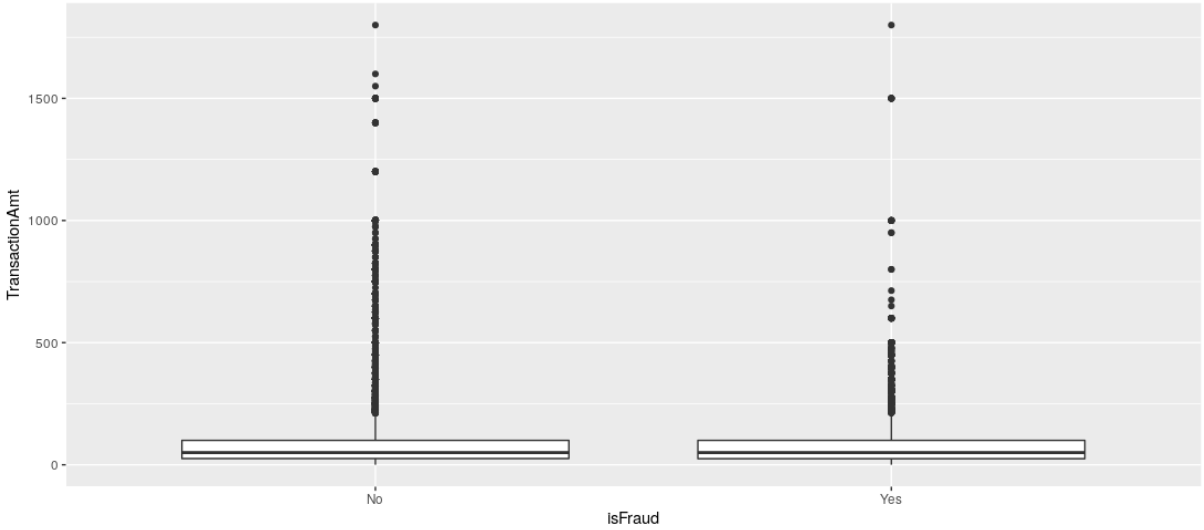
Figura 1: Distribución de la clase

Como puede observare la distribución de la clase esta muy desbalanceada, teniendo que más del 92 % de los casos son no fraudulentos. Debido a ello más tarde se equilibrarán las clases, realizando un sobremuestreo de la clase minoritaria, para no perder información de la clase minoritaria.

Veamos ahora la distribución de la variable TransactionAmt con respecto a la variable isFraud. El atributo TransactionAmt nos muestra la cantidad de dinero pagada en dolares en la transacción.

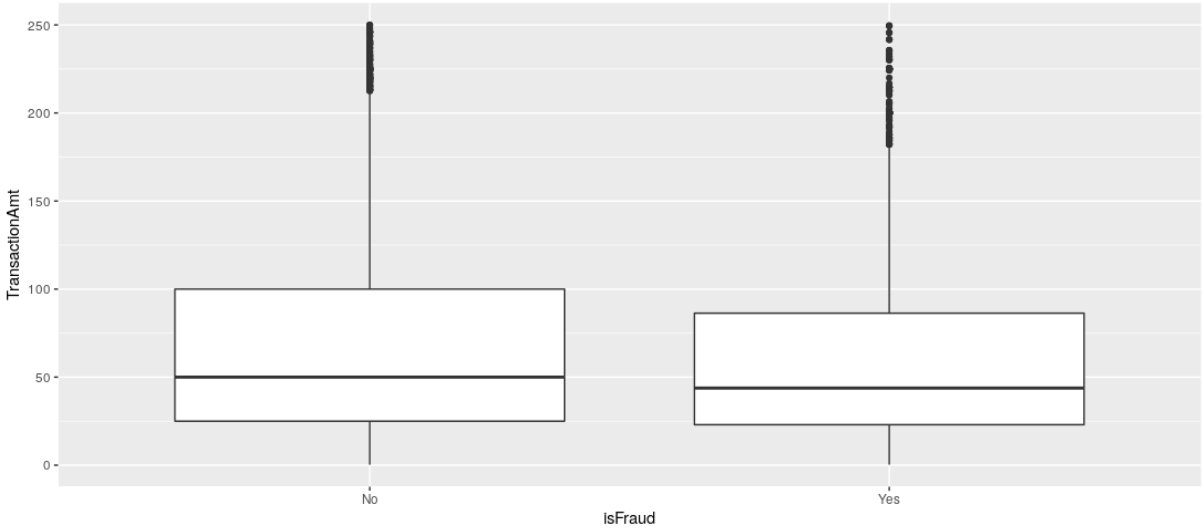
Esta distribución la vamos a visualizar mediante diagramas de cajas, los cuales nos permitirán visualizar fácilmente valores atípicos.

```
ggplot(df, aes(x=isFraud, y=TransactionAmt)) +
  geom_boxplot()
```



Vemos que la inmensa mayoría de las transacciones se sitúa por debajo de los 250 dólares, tanto para las transacciones fraudulentas, como para las legítimas. Los valores que se sitúan por encima de 250 se pueden considerar valores atípicos y representan una minoría de los casos. Debido a ello para poder visualizar mejor la distribución vamos a filtrar los datos y a quedarnos solamente con aquellas filas que estén por debajo de los 250 dólares en la transacción.

```
df_filter <- subset(df, TransactionAmt < 250)
ggplot(df_filter, aes(x=isFraud, y=TransactionAmt)) +
  geom_boxplot()
```



De la imagen anterior podemos apreciar que las transacciones fraudulentas son en general ligeramente inferiores a las no fraudulentas. Lo cual aporta a esta variable cierta capacidad predictiva.

La distribución de la variable TransactionAmt, nos indica que hay algunos valores atípicos, con lo cual podría ser interesante eliminar las filas que posean dichos valores atípicos, filtrando aquellas filas que tengan un valor inferior a 250 para dicho atributo.

Variables categóricas

En este subapartado vamos a analizar la frecuencia de algunas variables categóricas del dataset.

Empecemos con la variable P_emaildomain (la variable R_emaildomain no la mostraremos ya que los resultados son similares a esta).

```
summary(df$P_emaildomain)
```

```
        aim.com    anonymous.com          aol.com          att.net
             47            17840             4306              577
  bellsouth.net     cableone.net centurylink.net       cfl.rr.com
            501               27               16               37
    charter.net      comcast.net          cox.net    earthlink.net
            234             2196              515              122
 embarqmail.com     frontier.com frontiernet.net            gmail
             70               51               30              100
      gmail.com           gmx.de    hotmail.co.uk      hotmail.com
          54140              147              105            25782
     hotmail.de       hotmail.es       hotmail.fr       icloud.com
             43              270              290             1022
       juno.com         live.com      live.com.mx          live.fr
             56              737              653               55
        mac.com         mail.com           me.com          msn.com
            304              120              633             1069
    netzero.com      netzero.net    optonline.net      outlook.com
             11               10              265             2346
     outlook.es   prodigy.net.mx   protonmail.com          ptd.net
            396              165               36               25
          q.com   roadrunner.com   rocketmail.com    sbcglobal.net
             34               56               71              724
       sc.rr.com servicios-ta.com   suddenlink.net          twc.com
              9               35               40               38
    verizon.net           web.de   windstream.net      yahoo.co.jp
            758              237               46               32
    yahoo.co.uk        yahoo.com     yahoo.com.mx         yahoo.de
             44            11667             1182               70
       yahoo.es         yahoo.fr        ymail.com             NA's
             54              132              264            13391
```

Como podemos apreciar algunas de las variables que más se repiten son yahoo.com, gmail.com o hotmail.com. Sin embargo podemos ver que también hay otros dominios del mismo grupo a los mencionados como por ejemplo de yahoo tenemos yahoo.com, yahoo.es, yahoo.fr, etc... También tenemos algunos valores perdidos para esta variable, en concreto tenemos 13391 valores na. Estos datos nos sugieren realizar ciertas modificaciones en este atributo como por ejemplo agrupar los

dominios similares en uno solo como en el caso de yahoo, y también modificar el valore de los valores na por una etiqueta como po ejemplo 'Unknown'.

Ahora analicemos la variable id_31, la cual contiene el nombre del explorador web utilizado para llevar a cabo la transacción.

```
summary(df$id_31)
```

```
            chrome 63.0                mobile safari 11.0
                  22000                             13423
  mobile safari generic              ie 11.0 for desktop
                  11474                              9030
         safari generic                     chrome 62.0
                   8195                              7182
            chrome 65.0                     chrome 64.0
                   6871                              6711
chrome 63.0 for android                  chrome generic
                   5806                              4778
            chrome 66.0                      edge 16.0
                   4264                              4188
chrome 64.0 for android          chrome 65.0 for android
                   3473                              3336
           firefox 57.0                mobile safari 10.0
                   3315                              2779
chrome 66.0 for android          chrome 62.0 for android
                   2349                              2097
             edge 15.0       chrome generic for android
                   1600                              1158
           firefox 59.0              samsung browser 6.2
                   1099                              1061
           firefox 58.0                     chrome 49.0
                    833                               719
                firefox                ie 11.0 for tablet
                    673                               647
            chrome 61.0                     safari 11.0
                    642                               550
      mobile safari 9.0          chrome 61.0 for android
                    541                               538
    samsung browser 6.4                           chrome
                    509                               428
              edge 14.0                    firefox 52.0
                    419                               394
            chrome 60.0          chrome 60.0 for android
                    371                               325
                  other          chrome 55.0 for android
                    312                               302
    android webview 4.0          chrome 58.0 for android
                    285                               262
chrome 56.0 for android          chrome 59.0 for android
                    238                               227
           firefox 60.0              chrome 63.0 for ios
                    225                               222
samsung browser generic                     chrome 58.0
                    210                               188
           firefox 48.0              chrome 66.0 for ios
                    179                               159
chrome 52.0 for android             Samsung/SM-G532M
                    152                               150
```

```
        chrome 50.0 for android                     opera 49.0
                            145                            138
                    chrome 59.0                   firefox 56.0
                            133                            126
                    safari 10.0                firefox generic
                            111                            110
            samsung browser 7.0            android browser 4.0
                            101                            100
                    chrome 56.0        mobile safari uiwebview
                             98                             97
                    chrome 57.0        chrome 57.0 for android
                             94                             87
                          opera        chrome 54.0 for android
                             83                             82
            Generic/Android 7.0            chrome 65.0 for ios
                             81                             79
                  opera generic                   edge 17.0
                             78                             74
        chrome 43.0 for android        chrome 46.0 for android
                             72                             69
                    chrome 55.0                   edge 13.0
                             64                             62
            chrome 62.0 for ios                           edge
                             61                             58
                Generic/Android                 chrome 51.0
                             57                             56
                             ie google search application 48.0
                             56                             55
google search application 49.0                   opera 52.0
                             55                             55
              mobile safari 8.0        chrome 49.0 for android
                             54                             53
                Samsung/SM-G531H                  opera 51.0
                             52                             49
            chrome 64.0 for ios                   opera 53.0
                             46                             46
        chrome 51.0 for android                   safari 9.0
                             45                             43
            samsung browser 5.4        chrome 53.0 for android
                             43                             42
            samsung browser 5.2                firefox 47.0
                             41                             37
            samsung browser 4.0                       google
                             37                             36
                    chrome 67.0             Microsoft/Windows
                             33                             25
                  firefox 55.0                           silk
                             20                             19
                       (Other)                          NA's
                            135                           3951
```

Al igual que ocurría en el dominio del email, donde había grupos similares como el ejemplo de yahoo, aquí también ocurre. Por ejemplo podemos ver que tenemos numerosas versiones de chrome o de firefox. Esto nos vuelve a sugerir que podría ser buena idea agrupar aquellos navegadores web con cadenas similares, por ejemplo modificado el valor de la cadena de 'chrome -version' a la cadena 'chrome' y de esta forma reducir el número de nivéles en el factor.

No vamos a mostrar los datos para no alargar este apartado, pero en el caso de la variable

DeviceInfo vemos que ocurre lo mismo, presentando por ejemplo varias versiones de móviles Samsung o de Windows.

Algo que se puede observar en las variables analizadas es que hay ciertos valores que se repiten muy pocas veces, todos estos podemos agruparlos en un solo grupo etiquetado como 'Others' y así reducimos aún más los niveles de los factores.

Matriz de correlación

La matriz de correlación nos indica la correlación existente entre las variables numéricas. Si el valor es cercano a 0 significa que no hay correlaciones entre las variables, en cambio tanto si el valor se acerca a +1, como a -1, significará que existe una correlación entre las variables. Si esta correlación es positiva no indicará que cuando una de las variables crece la otra también lo hace. Si la correlación es negativa nos indicará que cuando una de las variables crece la otra decrece. Cuando más cercano a 1 en valor absoluto, sea este valor indicara una relación mayor en este sentido.

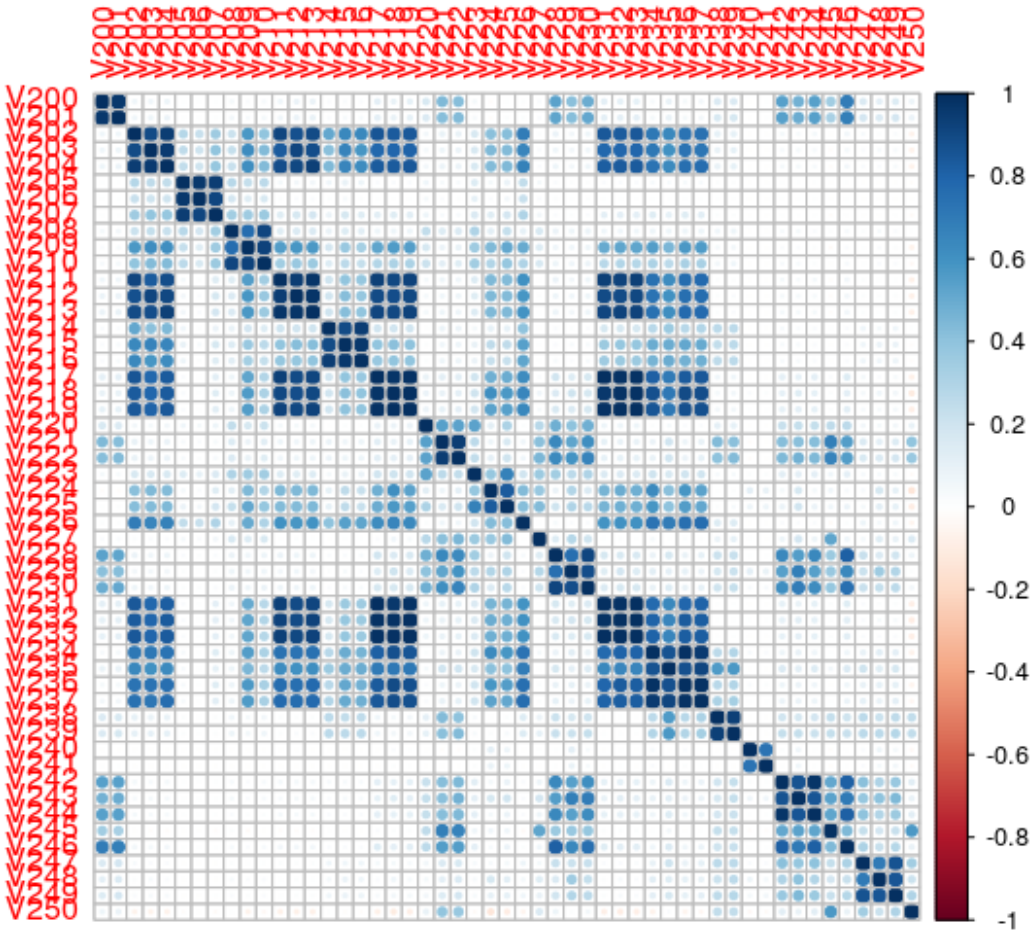
Para poder obtener la matriz de correlación de nuestros datos tenemos que tener 3 cosas en cuenta. La primer es que solo se puede obtener para variables numéricas, la segunda se trata de que no puede haber valores perdidos en dichas variables y la última cosa a tener en cuenta es que no puede haber una columna con desviación típica igual a 0, ya que en el proceso de obtención de la matriz de correlación se divide por dicho valor. Tener desviación típica igual a 0 es equivalente a que una variable siempre tenga el mismo valor. Por estos tres motivos necesitamos preparar los datos antes de obtener la matriz de correlación:

```
df <- df[, which(colMeans(is.na(df)) <= 0.4)]
df <- na.omit(df)
df <- Filter(function(x) length(unique(x))>1, df)
```

En el código anterior se han borrado algunas columnas que presentaban muchos valores NA, se han borrado también las filas que seguían teniendo valores NA tras borras las columnas y se han eliminado las variables en las que se repetía siempre el mismo valor.

Ahora ya estamos en condiciones de visualizar la matriz de correlación, debido a que está es demasiado grande, como causa de la alta dimensionalidad de nuestro dataset, solo se va a mostrar la matriz de correlación de un subconjunto de atributos.

```
# Obtener matriz de correlacion
cor_matrix <- cor(df[paste("V", 200:250, sep="")])
# Visualizar matriz de correlacion
corrplot(cor_matrix)
```



En la imagen anterior podemos visualizar la matriz de correlación para la variables V200 - V250. Gracias al color podemos distinguir fácilmente que variables presentan una alta correlación, de hecho podemos ver que el color azul oscuro está presente en varias casillas, lo cual nos indica una alta correlación positiva. En la diagonal evidentemente tenemos correlación igual a 1, ya que una variable consigo misma tiene una correlación perfecta.

El que haya variables altamente correladas entre sí, nos puede indicar que hay información redundante, ya que si ambas se comportan de la misma forma, pueden estar aortandonos la misma información.

3. Preprocesamiento de datos

En esta sección se explicará el preprocesamiento de datos realizado.

Convertir variables a categóricas

En primer lugar se ha procedido a convertir algunas variables que aparecen como numéricas a factores, ya que en realidad son variables categóricas. Como muchas de estas son variables que tiene el mismo nombre y solo varían en el número final como por ejemplo card1, card2, ..., card6, se ha implementado una función a la que se le pasa la cadena inicial y el rango, en este caso la cadena sería 'card' y el rango 1:6.

```
column_range_to_factor <- function(df, col, range) {
  for(i in range){
    col_name <- paste(col, i, sep = "")
    df[,col_name] <- as.factor(df[,col_name])
  }
  return(df)
}
```

Ahora usamos dicha función para convertir a factor las variables que queremos:

```
# Convertir a factor
df <- df %>%
    column_range_to_factor('card', 1:6) %>%
    column_range_to_factor('addr', 1:2) %>%
    column_range_to_factor('M', 1:9) %>%
    column_range_to_factor('id_', 12:38)
```

Supresión de columnas y tratamiento de valores perdidos

Ahora vamos a eliminar dos columnas: TransactionID y TransactionDT. TransactionID la borramos ya que es un identificador de la transacción y este es único para cada una de las transacciones por lo que no nos va a aportar ninguna información. La variable TransactionDT contiene un valor que representa el segundo en el que se realizó la transacción y se va a eliminar porque conocer el segundo en el que se realizó una transacción no aporta nada de información. Es cierto que se podría haber obtenido cierta información esta variable como por ejemplo el día de la semana o las horas, pero por problemas de tiempo se ha optado por no hacerlo y simplemente quitar la variable.

```
# Eliminar columnas TransactionID y TrnasactionDT
df <- subset(df, select = -c(TransactionID, TransactionDT))
```

Algo que vimos en el análisis exploratorio de datos es que había muchas columnas con un

gran porcentaje de valores perdidos, debido a ello e ha decidido eliminar aquellas variables que presentan un valor superior a un 40 % de valores na. De esta forma se consigue una reducción significativa de la dimensionalidad del dataset.

```
# Eliminar columnas con mas de un 40% de valores perdidos
df <- df[, which(colMeans(is.na(df)) <= 0.4)]
```

Ahora vamos a realizar el tratamiento de las variables que quedan con valores perdidos.

Vamos a comenzar tratando los valores perdidos de las variables categóricas. Una de las opciones que tendríamos sería reemplazar el valor NA por el valor más repetido, pero de esta forma estaríamos falseando los datos y perdiendo a información de que cierta variable tiene un valor perdido para algunas filas. El problema es que muchos algoritmos y funciones de R, no funcionan si hay valores NA. Por ello lo que vamos a hacer es sustituir los valores NA por la cadena 'Unknown'.

```
# Reemplazar Na por 'Unknown' en variables categ ricas
df<-df %>% mutate_if(is.factor, fct_explicit_na, na_level = 'Unknown')
```

Para el caso de las numéricas no tenemos la opción anterior. Algunas de las opciones que tenemos son sustituir por la media o por la mediana. Otra opción podría ser eliminar aquellas filas que tengan valores perdidos, pero des esta forma podríamos estar perdiendo mucha información, por ello antes de tomar ninguna decisión se ha comprobado cuantas filas perderíamos si hacemos esta última opción.

```
# Calcular la diferencia de filas si borramos filas con NA
nrow(df)-nrow(na.omit(df))
```

```
[1] 18415
```

Como podemos ver, en comparación con el tamaño del dataset no son tantas filas las que perdemos, debido a ello se ha optado finalmente por eliminar dichas filas.

```
# Eliminar filas con valores NA
df <- na.omit(df)
```

Durante el análisis exploratorio de los datos, vimos al ejecutar la función df_status, que algunas variables tenían el valor 1 para la columna **unique**, esto significa que en dichos atributos solo se presenta un posible valor, luego estas variables no nos aportan ninguna información y por ello se ha procedido a borrarlas.

```
df <- Filter(function(x) length(unique(x))>1, df)
```

A veces tenemos variables que presentan una alta correlación entre ellas, esto significa que ambas

variables nos aportan la misma información y debido a ello podríamos permitirnos prescindir de alguna de ellas. Precisamente eso es lo que vamos a hacer y vamos a eliminar variables que tengan una correlación en valor absoluto, superior a 0,9, quedándonos solo con una de ellas. Para ello nos vamos a ayudar de la función `findCorrelation`, que podemos encontrar en le paquete de **caret**. Esta función nos devuelve un vector de enteros con los índices de las columnas a borrar.

Para logra esto se ha implementado la siguiente función:

```
remove_correlated_columns <- function(df, correlation_value) {
  # Obtenemos columnas num ricas
  df_numeric <- df[, unlist(lapply(df, is.numeric))]
  # Obtenemos la matriz de correlaci n
  cor_matrix <- cor(df_numeric)

  hc <- findCorrelation(cor_matrix, cutoff=correlation_value)
  hc <- sort(hc)

  # Obtenemos los nombres d ela columnas que vamos a borras
  cols_to_remove <- colnames(df_numeric[, c(hc)])

  # Devolvemos el dataframe con las columnas borradas
  return(df[ , !names(df) %in% cols_to_remove])
}
```

Para eliminar las columnas altamente correladas solo tendríamos que hacer uso de la función definida:

```
df <- remove_correlated_columns(df, 0.9)
```

El valor de 0,9 indica que se van a eliminar las columnas con una correlación de más de 0,9 en valor absoluto.

También se observo durante el análisis exploratorio de datos que algunas columnas tenían un gran porcentaje de ceros, debido a ello se ha optado por eliminar aquellas columnas con un porentaje de eros superior al 90 %.

```
my_df_status <- df_status(df)
vars_to_remove=subset(my_df_status, my_df_status$p_zeros > 90)
df <- df[, !(names(df) %in% vars_to_remove[,"variable"])]
```

Reducción de niveles de factores

Ahora vamos a proceder a reducir los niveles de algunos factores. Como ya se vió en el análisis exploratorio de datos, algunas variables como por ejemplo P_emaildomain, tenían cadenas similares que solo se diferenciaban en ciertas terminaciones, como por ejemplo yahoo.com, yahoo.es, yahoo.fr. Debido a ello vamso a agrupar las cadenas que sean similares y además vamos a crear

una nueva clase denominada 'Other', para incluir en dicho grupo aquellos valores que aparecían con poca frecuencia.

Para conseguir esto vamos a hacer uso de la función **grepl**, la cual devuelve true si la cadena pasada como primer argumento se encuentra dentro de la segunda candena.

```
df <- df %>%
  mutate(id_31 = as.factor(
    case_when(
      grepl('chrome', id_31) ~ 'chrome',
      grepl('safari', id_31) ~ 'safari',
      grepl('edge', id_31) ~ 'edge',
      grepl('firefox', id_31) ~ 'firefox',
      grepl('ie', id_31) ~ 'ie',
      grepl('samsung', id_31) ~ 'samsung',
      TRUE ~ 'other'))) %>%
  mutate(DeviceInfo = as.factor(
    case_when(
      grepl('Windows', DeviceInfo) ~ 'Windows',
      grepl('iOS Device', DeviceInfo) ~ 'iOS Device',
      grepl('Unknown', DeviceInfo) ~ 'Unknown',
      grepl('MacOS', DeviceInfo) ~ 'MacOS',
      grepl('Trident', DeviceInfo) ~ 'Trident',
      grepl('SM', DeviceInfo) ~ 'Samsung',
      TRUE ~ 'Other'))) %>%
  mutate(P_emaildomain = as.factor(
    case_when(
      grepl('gmail', P_emaildomain) ~ 'gmail',
      grepl('hotmail', P_emaildomain) ~ 'hotmail',
      grepl('anonymous', P_emaildomain) ~ 'anonymous',
      grepl('Unknown', P_emaildomain) ~ 'Unknown',
      grepl('aol', P_emaildomain) ~ 'aol',
      grepl('outlook', P_emaildomain) ~ 'outlook',
      grepl('msn', P_emaildomain) ~ 'msn',
      grepl('live', P_emaildomain) ~ 'live',
      TRUE ~ 'Other'))) %>%
  mutate(R_emaildomain = as.factor(
    case_when(
      grepl('gmail', R_emaildomain) ~ 'gmail',
      grepl('hotmail', R_emaildomain) ~ 'hotmail',
      grepl('anonymous', R_emaildomain) ~ 'anonymous',
      grepl('Unknown', R_emaildomain) ~ 'Unknown',
      grepl('aol', R_emaildomain) ~ 'aol',
      grepl('outlook', R_emaildomain) ~ 'outlook',
      grepl('msn', R_emaildomain) ~ 'msn',
      grepl('live', R_emaildomain) ~ 'live',
      TRUE ~ 'Other')))
```

Tratamiento de outliers

Como valores atípicos solo se observaron estos para el caso de la variable TransactionAmt, debido a que hay una gran cantidad de variables y era inviable analizarlas todas. Como pudimos ver la

inmensa mayoría de los casos tenía una valor por debajo de 250, por ello se ha decidido eliminar aquellas filas que tengan un valor superior a 250 para dicha variable.

```
# Seleccionar filas con un valor inferior a 250
df <- subset(df, TransactionAmt <= 250)
```

Sobremuestreo

Para solucionar el problema del desbalanceo de los datos con respecto a la variable **isFraud** que utilizaremos para clasificar, se ha procedido a realizar un sobremuestremo del conjunto de datos, para crear nuevos ejemplo de la clase minoritaria. Para ello se ha utilizado la librería ROSE.

```
library("ROSE")
```

Dicha librería posee una función denominada **ovun.sample**, que sirve equilibrar conjuntos de datos con respecto a una variable. Uno de sus argumentos es **method**, en cual debemos de especificar el método que queremos utilizar para resolver el desbalanceo de datos, en nuestro caso elegiremos el método **over**, el cual se corresponde con el sobremuestreo.

```
df <- ovun.sample(isFraud~.,data = df, method = "over")$data
```

Mencionar que también se probó el método SMOTE, el cual hace uso de KNN para determinar vecinos cercanos y crear ejemplos a partir de ellos, pero finalmente desistí ya que tardaba demasiado en ejecutar tuve que detenerlo.

4. Selección de variables

Además del preprocesamiento realizado anteriormente, se va a realizar una selección de las variables más relevantes para la clasificación. Para ello vamos a recurrir al uso de random forest, el cual tras entrenar el modelo nos va a permitir obtener las importancias que ha asignado a cada variable.

Para ello vamos a comenzar leyendo los datos que hemos preprocesado.

```
df <- read.csv(file = "./data/data_train_preprocessing.csv", header =
    TRUE, sep  = ",")
```

Antes de saber nada de lo que pasaría, utilice dicho dataset para entrenar un modelo de ranfom forest, pero tuve que detenerlo porque iba demasiado lento. Pensé que tal vez el motivo de que fuera tan lento, fuese el que había en concreto 7 variables categóricas con muchos valores posibles, de hecho una de ellas superaba los 8000 valores posibles como pude comprobar tras ejecutar la función **df_status**. Creo que ese era el motivo así que el siguiente paso que realice, fue el de eliminar aquellas variables categóricas con más de 20 valores posibles:

```
df <- Filter(function(x){(is.factor(x) & length(unique(x))< 20) | !is.
    factor(x)}, df)
```

Tras esto volví a ejecutar y apenas tardó 15 minutos en ejecutar el algoritmo de entrenamiento.

```
rfmodel <- randomForest(x = df[,2:92], y = df$isFraud, importance = T,
     ntree=100)
```

Tras entrenar el modelo podemos obtener la importancia de cada variable de la siguiente forma:

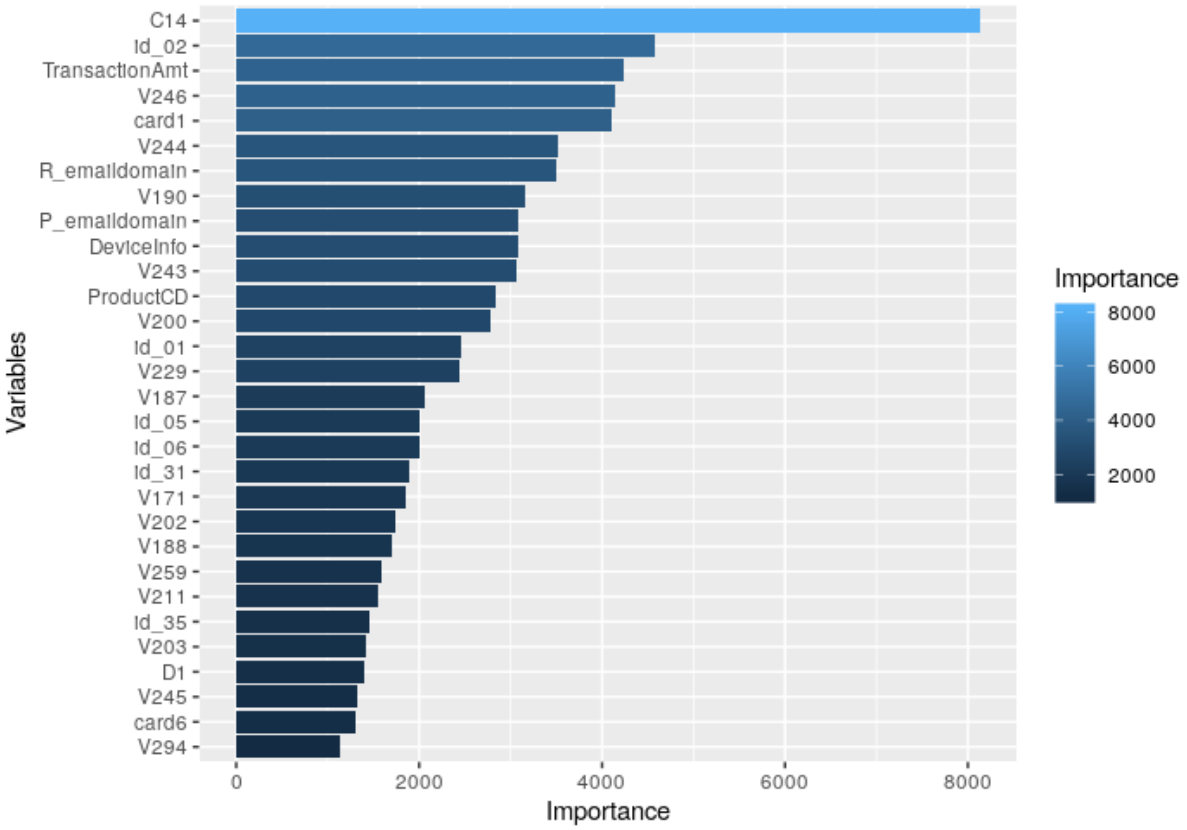
```
# Obtener importancia de las variables
importance <- importance(rfmodel)

varImportance <- data.frame(Variables = row.names(importance),
                            Importance = round(importance[ ,'
                               MeanDecreaseGini'],2))
```

Ahora que tenemos los valores de importancia de las variables, procedemos a ordenarlos de forma de creciente y nos quedamos con las 30 primeras variables.

```
# Ordenamos decrecientemente
varImportanceSorted <- varImportance[order(-varImportance$Importance)
    ,]
# Seleccionamos las 30 mas importantes
mostImportantVars <- varImportanceSorted[1:30,]
```

Una vez tenemos las variables más importantes las visualizamos en una gráfica:

```
ggplot(mostImportantVars, aes(x = reorder(Variables, Importance),
                              y = Importance, fill = Importance)) +
  geom_bar(stat='identity') +
  labs(x = 'Variables') +
  coord_flip()
```



5. Modelos de predicción

5.1. Regresión logística

El primer modelo que se ha utilizado es el de regresión logística, ya que es uno de los modelos más simples y nos dará una primera aproximación al problema.

Antes de empezar entrenar el modelo se han separado los datos de train, en dos conjuntos con uno de ellos entrenaremos y con el otro validaremos el modelo obtenido. Para realizar el entrenamiento se ha hecho una separación de forma estratificada de forma que se mantiene la proporción de clases en ambos conjuntos.

```
train.index <- createDataPartition(df$isFraud, p = .7, list = FALSE)
data_train <- df[ train.index,]
data_test  <- df[-train.index,]
```

Ahora entrenamos el modelo del regresión logística, al cual le debemos de aportar una fórmula indicando las variables que se usarán para la regresión. Las variables que se han indicado en la fórmula son algunas de las variables numéricas más importantes que obtuvimos en la selección de variables con random forest.

```
mylogit <- glm(isFraud ~ TransactionAmt + C14 + id_02 + V246 + V243 +
    V200 + V244 + id_01 + V229, data = data_train, family = "binomial")
```

Una vez entrenado el modelo, realizamos las predicciones sobre el conjunto de entrenamiento y obtenemos los valores de algunas medidas de rendimiento.

```
Confusion Matrix and Statistics

   predict
       0     1
  0 30479  3057
  1 13897 19710

               Accuracy : 0.7475
                 95% CI : (0.7442, 0.7508)
    No Information Rate : 0.6609
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4952

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6868
            Specificity : 0.8657
         Pos Pred Value : 0.9088
         Neg Pred Value : 0.5865
             Prevalence : 0.6609
         Detection Rate : 0.4539
   Detection Prevalence : 0.4995
      Balanced Accuracy : 0.7763
```

```
        'Positive' Class : 0
```

5.2. Random Forest

En este apartado vamos a entrenar un modelo de random forest. Para ello al igual que hicimos en la regresión logística vamso a separar en train y test un conjunto de datos ya con las variables que seleccionamos con las importancias obtenidas por random forest. Para entrenar el modelo simplemente debemos ejecutar lo siguiente:

```
rfmodel <- randomForest(x = data_train[,2:31],
                        y = data_train$isFraud,
                        importance = T,
                        do.trace=TRUE,
                        ntree=200)
```

Una vez el modelo ha terminado de entrenar realizamos las predicciones sobre el conjunto de test.

```
predTest <- predict(rfmodel, data_test, type = "class")
```

```
table(data_test$isFraud, predTest)
mean(predTest == data_test$isFraud)
```

```
    predTest
       Yes    No
  Yes  33522 891
  No      85  32645

  [1] 0.9854639
```

Como podemos observar los resultados con random forest han sido notablemente mejores. A pesar de que los datos estaban desbalanceados se ha conseguido una buena tasa de acierto en el caso de fraudes, y una tasa baja de falso positivos y falsos negativos, como puede apreciarse en la matriz de confusión.