Tipología y ciclo de vida de los datos: Práctica 2

Iván García Jiménez, Itziar Ricondo

5 de enero de 2021

1 Tareas a realizar en la práctica

Siguiendo las principales etapas de un proyecto analítico, las diferentes tareas a realizar (y justificar) son las siguientes:

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

2. Integración y selección de los datos de interés a analizar.

3. Limpieza de los datos.

3.1. ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

3.2. Identificación y tratamiento de valores extremos.

4. Análisis de los datos.

4.1. Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

4.2. Comprobación de la normalidad y homogeneidad de la varianza.

4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

5. Representación de los resultados a partir de tablas y gráficas.

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

7. Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.

2 Descripción del dataset.

¿Por qué es importante y qué pregunta/problema pretende responder?

2.1 Sobre el conjunto de datos

Este conjunto de datos contiene datos de uso de camiones de gran tonelaje de Scania. El caso de estudio se centra en el Sistema de Aire a Presión (APS, del inglés *Air Pressure System*), sistema responsable de proveer de aire comprimido a varias de las funciones del camión, como el freno o embrague. Los casos con clase positiva son fallos de componentes relacionados con el sistema APS. Por el contrario, los caos con clase negativa corresponden a fallos de componentes no relacionados con el APS. El conjunto de datos presenta casos de ambos tipos, si bien es cierto que los casos negativos son mucho más frecuentes.

El objetivo del conjunto de datos es predecir qué tipo de fallos tienen su origen en el sistema APS y cuáles son debidos a otras causas. Este hecho tiene repercusión en las revisiones o reparaciones a realizar sobre el camión. De hecho, la bondad de la predicción se medirá en términos de coste. El coste total se evaluará como la suma de fallos de tipo 1 por el coste asociado (Coste1) más la suma de fallos de tipo 2 por el coste asociado (Coste2).Un error de tipo I se origina cuando el modelo predice positivo cuando en realidad el fallo no tiene origen en un componente el APS. El coste 1 asociado a este hecho es debido a la revisión innecesaria del APS que debe ser realizada en el taller, que se ha valorado en 10 unidades. Por el contrario, el error de tipo II podría tener como resultado permitir el tráfico del camión con defecto en un componente APS, que puede causar una averís. El coste 2 asociado a este tipo de fallo es superior y se ha valorado en 500 unidades. El mejor modelo de predicción será aquel que minimice el coste total.

Este conjunto ha sido proporcionado por la empresa Scania y está disponibles en el repositorio UCI Machine Learning Repository [1] (https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks) (https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks (https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks)). Los datos proporcionados son un conjunto de datos de entrenamiento y otro de validación.

Este conjunto de datos trata sobre la predicción de fallos. Se ha seleccionado este conjunto de datos porque la identificación del estado de componentes (desgaste, fallo, vida remanente) es un tema de gran relevancia dentro de las iniciativas de Industria 4.0 y, en concreto, de técnicas de análisis de datos, minería y aprendizaje automático.

Por otro lado, este conjunto de datos presenta una serie de retos para los autores de este análisis, que se listan a continuación:

- Gran número de variables y además las variables están anonimizadas. El conjunto de datos de entrenamiento presenta 171 variables y 60000 observaciones. El hecho de que se hayan recodificado (anonimizado) las variables impide tener información física adicional que ayude en la interpretación. Sí se menciona el hecho de que algunas variables son atributos y otras representan histogramas. El número tan elevado de variables implica que **se tendrán que utilizar herramientas que permitan automatizar el tratamiento de los datos**, que supone un hecho diferencial sobre el tipo de tratamientos realizados hasta ahora por nosotros.
- Los casos positivos y negativos están desequilibrados. Deberá analizarse qué implica este hecho en el análisis de los datos.
- El conjunto de datos tiene un fuerte carácter industrial. El resultado debe ayudar en la toma de decisiones desde una perspectiva de coste.

3 Integración y selección de los datos de interés a analizar

3.1 Lectura del archivo de datos:

El dataset está compuesto por 2 archivos: aps_failure_training_set.csv que contiene los datos para entrenar el modelo, y aps_failure_test_set.csv que contiene los datos para evaluarlo.

Debido a la gran cantidad de datos del dataset, se ha optado por leerlos mediante la función “fread” de la librería “data.table” [2] (https://www.r-bloggers.com/2019/06/how-data-tables-fread-can-save-you-a-lot-of-time-and-memory-and-take-input-from-shell-commands/) , ya que esta función permite una lectura de datos más rápida que con la función “read.csv”.

En la descripción del dataset de UCI Machine Learning Repository, se nos informa de antemano que el dataset contiene valores nulos (perdidos) asignados como “na”, lo indicamos por parámetro en la función fread para que capte de manera correcta los valores perdidos.

```
if (!require("data.table")) install.packages("data.table")
```

```
## Loading required package: data.table
```

```
library(data.table)
# Lectura de los dataset de train y test
url_train <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00421/aps_failure_training_set.csv"
url_test <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00421/aps_failure_test_set.csv"
rtrain <- fread(url_train, na.strings = "na", strip.white = TRUE)
rtest <- fread(url_test, na.strings = "na", strip.white = TRUE)
# Dimensiones del dataset
dim(rtrain)
```

```
## [1] 60000   171
```

Podemos ver que el conjunto de entrenamiento contiene una gran cantidad de datos, 60000 registros y 171 atributos; a continuación vemos su estructura.

```
# Estructura del dataset
str(rtrain)
```

```
## Classes 'data.table' and 'data.frame':   60000 obs. of  171 variables:
##  $ class : chr  "neg" "neg" "neg" "neg" ...
##  $ aa_000: int  76698 33058 41040 12 60874 38312 14 102960 78696 153204 ...
##  $ ab_000: int  NA NA NA 0 NA NA 0 NA NA 0 ...
##  $ ac_000: int  2130706438 0 228 70 1368 2130706432 6 2130706432 0 182 ...
##  $ ad_000:integer64 280 NA 100 66 458 218 NA 116 ...
##  $ ae_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ af_000: int  0 0 0 10 0 0 0 0 0 0 ...
##  $ ag_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ag_001: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ag_002: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ag_003: int  0 0 0 318 0 0 0 0 0 11804 ...
##  $ ag_004: int  37250 18254 1648 2212 43752 9128 1202 2130 458 684444 ...
##  $ ag_005: int  1432864 653294 370592 3232 1966618 701702 3766 142462 440704 326536 ...
##  $ ag_006: int  3664156 1720800 1883374 1872 1800340 1462836 1150 4227340 4398806 31586 ...
##  $ ag_007: int  1007684 516724 292936 0 131646 449716 0 1674278 2179182 0 ...
##  $ ag_008: int  25896 31642 12016 0 4588 39000 0 59718 144418 0 ...
##  $ ag_009: int  0 0 0 0 0 660 0 876 808 0 ...
##  $ ah_000: int  2551696 1393352 1234132 2668 1974038 1087760 2094 2738458 3209768 2658638 ...
##  $ ai_000: int  0 0 0 0 0 0 0 0 0 14346 ...
##  $ aj_000: int  0 68 0 0 226 0 0 0 80 0 ...
##  $ ak_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ al_000: int  0 0 0 642 0 0 0 0 0 29384 ...
##  $ am_0  : int  0 0 0 3894 0 0 0 0 0 46356 ...
##  $ an_000: int  4933296 2560898 2371990 10184 3230626 2283060 12460 5586558 6289144 5566182 ...
##  $ ao_000: int  3655166 2127150 2173634 7554 2618878 1892752 11044 4916376 5590784 4426834 ...
##  $ ap_000: int  1766008 1084598 300796 10764 1058136 469244 2292 1193648 1256914 1571480 ...
##  $ aq_000: int  1132040 338544 153698 1014 551022 347054 402 568900 579650 1120644 ...
##  $ ar_000: int  0 0 0 0 0 0 0 0 0 2 ...
##  $ as_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ at_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ au_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ av_000: int  1012 0 358 60 1788 1142 0 182 0 1368 ...
##  $ ax_000: int  268 0 110 6 642 452 0 34 0 862 ...
##  $ ay_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ay_001: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ay_002: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ay_003: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ay_004: int  0 0 0 0 42124 0 0 0 15198 0 ...
##  $ ay_005: int  469014 71510 0 0 372236 280112 0 34444 123920 0 ...
##  $ ay_006: int  4239660 772720 870456 0 2128914 1160742 0 1558518 898176 0 ...
##  $ ay_007: int  703300 1996924 239798 2038 819596 774914 0 3996204 1096132 94188 ...
##  $ ay_008: int  755876 99560 1450312 5596 584074 447274 6118 517638 5030950 960182 ...
##  $ ay_009: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ az_000: int  5374 7336 1620 64 1644 1580 1484 6634 5574 250 ...
##  $ az_001: int  2108 7808 1156 6 362 680 14 6406 1170 16 ...
##  $ az_002: int  4114 13776 1228 6 562 886 24 12848 2110 32 ...
##  $ az_003: int  12348 13086 34250 914 842 1270 16 34152 3372 146 ...
##  $ az_004: int  615248 1010074 1811606 76 30194 42450 88 427644 61402 126 ...
##  $ az_005: int  5526276 1873902 710672 2478 3911734 2615652 4492 5618652 7090548 951126 ...
##  $ az_006: int  2378 14726 34 2398 1606 524 0 468 174 91162 ...
##  $ az_007: int  4 6 0 1692 0 0 0 0 20 11512 ...
##  $ az_008: int  0 0 0 0 0 0 0 0 4 0 ...
##  $ az_009: int  0 0 0 0 0 0 0 0 2 0 ...
##  $ ba_000: int  2328746 1378576 790690 6176 1348578 1037372 4332 1699220 3982850 702994 ...
##  $ ba_001: int  1022304 447166 672026 340 1035668 710296 646 1550888 1580644 186172 ...
##  $ ba_002: int  415432 199512 332340 304 338762 244992 316 619780 567792 69282 ...
##  $ ba_003: int  287230 154298 254892 102 236540 145738 54 445704 293120 36640 ...
##  $ ba_004: int  310246 137280 189596 74 182278 99242 140 323882 231626 20698 ...
##  $ ba_005: int  681504 138668 135758 406 151778 70704 444 261558 372878 17720 ...
##  $ ba_006: int  1118814 165908 103552 216 163248 90386 122 337086 135466 18586 ...
##  $ ba_007: int  3574 229652 81666 16 470800 260950 16 868682 0 2278 ...
##  $ ba_008: int  0 87082 46 0 19292 3362 14 4 0 0 ...
##  $ ba_009: int  0 4708 0 0 0 0 34 0 0 0 ...
##  $ bb_000: int  6700214 3646660 2673338 21614 4289260 2752812 14764 6781222 7547346 7167172 ...
##  $ bc_000: int  0 86 128 2 448 474 0 16 0 692 ...
##  $ bd_000: int  10 454 202 12 556 2974 0 20 42 152 ...
##  $ be_000: int  108 364 576 0 642 502 0 236 50 1690 ...
##  $ bf_000: int  50 350 4 0 2 4 0 12 960 294 ...
##  $ bg_000: int  2551696 1393352 1234132 2668 1974038 1087760 2094 2738458 3209768 2658638 ...
##  $ bh_000: int  97518 49028 28804 184 86454 37720 124 78114 113418 126990 ...
```

```
##  $ bi_000: int  947550 688314 160176 7632 653692 250044 1074 872308 554154 355360 ...
##  $ bj_000: int  799478 392208 139730 3090 399410 218150 1206 319140 700112 1215418 ...
##  $ bk_000: int  330760 341420 137160 NA 306780 186500 NA 214220 194720 263620 ...
##  $ bl_000: int  353400 359780 130640 NA 282560 222800 NA 186520 202340 263960 ...
##  $ bm_000: int  299160 366560 NA NA 274180 NA NA 171320 222840 253220 ...
##  $ bn_000: int  305200 NA NA NA NA NA 214900 181400 260500 ...
##  $ bo_000: int  283680 NA NA NA NA NA 258980 187820 270840 ...
##  $ bp_000: int  NA NA NA NA NA NA NA 211420 NA ...
##  $ bq_000: int  NA NA NA NA NA NA NA NA NA ...
##  $ br_000: int  NA NA NA NA NA NA NA NA NA ...
##  $ bs_000: int  178540 6700 28000 10580 189000 55280 2200 137860 43560 255200 ...
##  $ bt_000: num  76698.1 33057.5 41040.1 12.7 60874 ...
##  $ bu_000: int  6700214 3646660 2673338 21614 4289260 2752812 14764 6781222 7547346 7167172 ...
##  $ bv_000: int  6700214 3646660 2673338 21614 4289260 2752812 14764 6781222 7547346 7167172 ...
##  $ bx_000: int  6599892 3582034 2678534 21772 4283332 2762528 15690 6800488 7544078 1065896 ...
##  $ by_000: num  43566 17733 15439 32 24793 ...
##  $ bz_000: int  68656 260120 7466 50 17052 21730 200 3180 186404 415352 ...
##  $ ca_000: int  54064 115626 22436 1994 61844 30254 1412 67254 20680 108024 ...
##  $ cb_000: int  638360 6900 248240 21400 654700 328460 14620 721460 276200 1114900 ...
##  $ cc_000: int  6167850 2942850 2560566 7710 3946944 2663042 6158 6106804 7164376 6785612 ...
##  $ cd_000: int  1209600 1209600 1209600 1209600 1209600 1209600 1209600 1209600 1209600 1209600 ...
##  $ ce_000: int  246244 0 63328 302 135720 74986 0 173620 0 0 ...
##  $ cf_000:integer64 2 NA 0 2 0 2 NA 2 ...
##  $ cg_000: int  96 NA 124 6 152 50 NA 116 NA NA ...
##  $ ch_000: int  0 NA 0 0 0 0 NA 0 NA NA ...
##  $ ci_000: num  5245752 2291079 2322692 2135 3565685 ...
##  $ cj_000: num  0 0 0 0 0 ...
##  $ ck_000: num  916568 643537 236100 4525 379112 ...
##  $ cl_000: int  6 0 0 2 0 2 0 142 2 0 ...
##   [list output truncated]
##  - attr(*, ".internal.selfref")=<externalptr>
```

3.2 Consideraciones iniciales sobre la estructura de los datos

Los datos son leídos como chart, hay que pasar a numeric. La primera variable del dataset es la clase, es decir, el valor a predecir. El resto de las variables son numéricas, pero algunas se han asignado como integer64, para evitar errores se ha decidido transformar las variables de tipo integer64 a tipo entero.

```
if (!require("dplyr")) install.packages("dplyr")
```

```
## Loading required package: dplyr
```

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

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

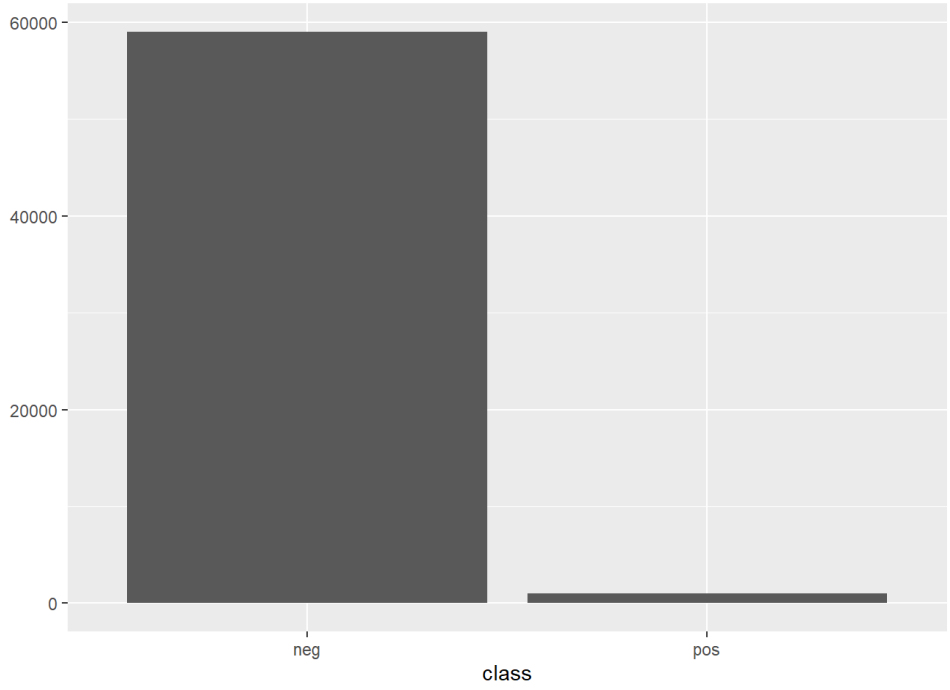
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```
library(dplyr)
is.integer64 <- function(x){class(x)=="integer64"}
rtrain <- rtrain %>% mutate_if(is.integer64, as.integer)
rtest <- rtest %>% mutate_if(is.integer64, as.integer)
```

Nos interesa ver cómo se distribuye la variable de clasificación, esto influye considerablemente tanto en la limpieza a realizar como en su análisis posterior. El 98.3% de los casos son negativos y el 12.7% positivos, esto es, existe un desequilibrio grande entre los datos.

```
# visualizamos la distribución de la variable class
library(ggplot2)
qplot(as.factor(rtrain$class), xlab = "class")
```




```
prop.table(table(rtrain$class))
```

```
##
##        neg        pos
## 0.98333333 0.01666667
```

Podemos observar un claro desequilibrio entre clases, tenemos muchos más camiones que han dado negativo (98.33%) que camiones que han dado positivo (16.66%). Este desequilibrio es algo que se debe tener en cuenta al tratar con este dataset.

Se hace un screening de datos mediante las funciones “ExpData”de la librería “SmartEDA” y “skim” de la libreria “skimr” [3] (https://www.datanovia.com/en/blog/display-a-beautiful-summary-statistics-in-r-using-skimr-package/). Se ha decidido utilizar estas funciones en vez de “summary”, ya que al tener grandes cantidades de datos, es más visual ver un atributo por fila tal y como vemos a continuación:

```
if (!require("SmartEDA")) install.packages("SmartEDA")
```

```
## Loading required package: SmartEDA
```

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

```
library(SmartEDA)
library(skimr)
## EDA, Datos faltantes, ceros y outliers
ExpData(data=rtrain,type=1)
```

```
##                                         Descriptions        Value
## 1                                   Sample size (nrow)        60000
## 2                              No. of variables (ncol)          171
## 3                      No. of numeric/interger variables          170
## 4                                No. of factor variables            0
## 5                                  No. of text variables            1
## 6                               No. of logical variables            0
## 7                             No. of identifier variables            0
## 8                                  No. of date variables            0
## 9               No. of zero variance variables (uniform)            1
## 10                 %. of variables having complete cases    1.17% (2)
## 11   %. of variables having >0% and <50% missing cases 94.15% (161)
## 12 %. of variables having >=50% and <90% missing cases    4.68% (8)
## 13          %. of variables having >=90% missing cases       0% (0)
```

```
eda_skim<-skim(rtrain)
eda_skim
```

Data summary

Name	rtrain
Number of rows	60000
Number of columns	171
_____	
Column type frequency:	
character	1
numeric	170
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
class	0	1	3	3	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
aa_000	0	1.00	59336.50	145430.06	0	834.00	30776.00	48668.00	2746564.0	▇▁▁▁▁
ab_000	46329	0.23	0.71	3.48	0	0.00	0.00	0.00	204.0	▇▁▁▁▁
ac_000	3335	0.94	356014263.03	794874918.48	0	16.00	152.00	964.00	2130706796.0	▇▁▁▁▂
ad_000	14862	0.75	445.91	3372.04	0	24.00	126.00	430.00	612266.0	▇▁▁▁▁
ae_000	2500	0.96	6.82	161.54	0	0.00	0.00	0.00	21050.0	▇▁▁▁▁
af_000	2500	0.96	11.01	209.79	0	0.00	0.00	0.00	20070.0	▇▁▁▁▁
ag_000	671	0.99	221.64	20478.46	0	0.00	0.00	0.00	3376892.0	▇▁▁▁▁
ag_001	671	0.99	975.72	34200.53	0	0.00	0.00	0.00	4109372.0	▇▁▁▁▁
ag_002	671	0.99	8606.01	150322.03	0	0.00	0.00	0.00	10552856.0	▇▁▁▁▁
ag_003	671	0.99	88591.28	761731.19	0	0.00	0.00	0.00	63402074.0	▇▁▁▁▁
ag_004	671	0.99	437096.63	2374281.89	0	308.00	3672.00	49522.00	228830570.0	▇▁▁▁▁
ag_005	671	0.99	1108373.83	3262607.27	0	13834.00	176020.00	913964.00	179187978.0	▇▁▁▁▁
ag_006	671	0.99	1657817.89	3909383.84	0	10608.00	930336.00	1886608.00	94020666.0	▇▁▁▁▁
ag_007	671	0.99	499309.80	1422764.51	0	0.00	119204.00	588820.00	63346754.0	▇▁▁▁▁
ag_008	671	0.99	35569.89	220152.35	0	0.00	1786.00	26690.00	17702522.0	▇▁▁▁▁
ag_009	671	0.99	5114.75	169658.17	0	0.00	0.00	364.00	25198514.0	▇▁▁▁▁

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ah_000	645	0.99	1809931.18	4185740.13	0	29733.00	1002420.00	1601366.00	74247318.0	
ai_000	629	0.99	9016.97	163277.85	0	0.00	0.00	0.00	16512852.0	
aj_000	629	0.99	1143.67	50359.71	0	0.00	0.00	0.00	5629340.0	
ak_000	4400	0.93	979.49	75831.62	0	0.00	0.00	0.00	10444924.0	
al_000	642	0.99	59130.48	539465.81	0	0.00	0.00	1204.00	34762578.0	
am_0	629	0.99	93281.33	849469.39	0	0.00	0.00	2364.00	55903508.0	
an_000	642	0.99	3461037.04	7790349.73	0	73238.50	1918629.00	3128415.50	140861830.0	
ao_000	589	0.99	3002440.31	6819518.39	0	65585.00	1643556.00	2675796.00	122201822.0	
ap_000	642	0.99	1004159.56	3088457.03	0	25189.00	357281.00	724660.50	77934944.0	
aq_000	589	0.99	442404.46	1262468.89	0	4161.00	178792.00	376900.00	25562646.0	
ar_000	2723	0.95	0.50	5.51	0	0.00	0.00	0.00	350.0	
as_000	629	0.99	126.74	11010.04	0	0.00	0.00	0.00	1655240.0	
at_000	629	0.99	5072.05	119615.94	0	0.00	0.00	0.00	10400504.0	
au_000	629	0.99	230.58	15799.52	0	0.00	0.00	0.00	2626676.0	
av_000	2500	0.96	1117.83	6598.61	0	12.00	116.00	646.00	794458.0	
ax_000	2501	0.96	374.33	1482.71	0	10.00	66.00	263.00	116652.0	
ay_000	671	0.99	12211.65	454496.33	0	0.00	0.00	0.00	50553892.0	
ay_001	671	0.99	10190.12	535270.68	0	0.00	0.00	0.00	80525378.0	
ay_002	671	0.99	10975.00	428336.96	0	0.00	0.00	0.00	28474838.0	
ay_003	671	0.99	7225.78	206467.88	0	0.00	0.00	0.00	13945170.0	
ay_004	671	0.99	10566.00	354625.82	0	0.00	0.00	0.00	40028704.0	
ay_005	671	0.99	111979.11	1394585.22	0	0.00	0.00	40132.00	124948914.0	
ay_006	671	0.99	1078551.20	3278941.40	0	0.00	168202.00	1270244.00	127680326.0	
ay_007	671	0.99	1546032.36	5106177.30	0	6118.00	348622.00	1337364.00	489678156.0	
ay_008	671	0.99	1051122.96	3991049.53	0	7524.00	94812.00	611816.00	104566992.0	
ay_009	671	0.99	1162.62	97960.80	0	0.00	0.00	0.00	18824656.0	
az_000	671	0.99	7849.61	73636.76	0	1028.00	2098.00	4150.00	10124620.0	
az_001	671	0.99	4420.99	33995.39	0	60.00	636.00	2016.00	4530258.0	
az_002	671	0.99	8066.08	106599.72	0	90.00	1016.00	3136.00	14217662.0	
az_003	671	0.99	87240.82	653256.15	0	294.00	3570.00	43014.00	45584242.0	
az_004	671	0.99	1476896.66	4184087.05	0	1542.00	81614.00	1774410.00	123047106.0	
az_005	671	0.99	2135583.86	6460220.91	0	38638.00	527034.00	1796172.00	467832334.0	
az_006	671	0.99	101894.26	899769.94	0	10.00	292.00	4218.00	64589140.0	
az_007	671	0.99	17377.82	280444.55	0	0.00	0.00	0.00	39158218.0	
az_008	671	0.99	661.79	14924.75	0	0.00	0.00	0.00	1947884.0	
az_009	671	0.99	42.07	3256.57	0	0.00	0.00	0.00	666148.0	
ba_000	688	0.99	1399651.96	3777091.37	0	33478.50	679486.00	1276368.00	232871714.0	
ba_001	688	0.99	894117.47	2346002.01	0	14833.50	443641.00	811000.50	116283282.0	
ba_002	688	0.99	413096.93	1196082.58	0	5207.50	186046.00	340408.50	55807388.0	
ba_003	688	0.99	274006.96	748538.76	0	1842.00	134149.00	244449.00	36931418.0	
ba_004	688	0.99	204875.58	539058.40	0	612.00	101874.00	197162.00	25158556.0	
ba_005	688	0.99	188941.19	509267.18	0	374.00	83992.00	184674.00	19208664.0	
ba_006	688	0.99	210628.82	637081.50	0	352.00	70116.00	204959.00	18997660.0	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ba_007	688	0.99	185787.39	525150.63	0	72.00	4470.00	207471.00	14314086.0	
ba_008	688	0.99	35882.84	243665.68	0	0.00	22.00	1820.00	31265984.0	
ba_009	688	0.99	35766.72	335327.61	0	0.00	0.00	60.00	43706408.0	
bb_000	645	0.99	4526177.19	10886736.63	0	105601.00	2360728.00	3868370.00	192871534.0	
bc_000	2725	0.95	569.53	4045.94	0	0.00	16.00	136.00	396952.0	
bd_000	2727	0.95	921.78	4833.07	0	8.00	66.00	438.00	306452.0	
be_000	2503	0.96	1372.65	9249.94	0	18.00	180.00	614.00	810568.0	
bf_000	2500	0.96	74.88	546.79	0	0.00	2.00	18.00	51050.0	
bg_000	642	0.99	1809430.98	4180199.78	0	29743.00	1002718.00	1602765.50	74247318.0	
bh_000	642	0.99	57943.08	152209.32	0	852.00	26352.00	49085.50	3200582.0	
bi_000	589	0.99	492207.57	1485184.51	0	15947.00	179842.00	379610.00	44937496.0	
bj_000	589	0.99	510089.23	1820104.91	0	8522.00	154404.00	333608.00	45736316.0	
bk_000	23034	0.62	280429.11	261301.48	0	162720.00	210660.00	281115.00	1310700.0	
bl_000	27277	0.55	321353.69	319210.98	0	170540.00	222540.00	303150.00	1310700.0	
bm_000	39549	0.34	399603.17	407071.85	0	172210.00	239140.00	369100.00	1310700.0	
bn_000	44009	0.27	463710.83	464447.34	0	171720.00	251400.00	493100.00	1310700.0	
bo_000	46333	0.23	513147.82	497353.67	0	170550.00	270660.00	1310700.00	1310700.0	
bp_000	47740	0.20	551389.80	519611.45	0	172170.00	288320.00	1310700.00	1310700.0	
bq_000	48722	0.19	582871.32	536697.03	0	170420.00	305100.00	1310700.00	1310700.0	
br_000	49264	0.18	604886.61	547227.87	0	169470.00	320400.00	1310700.00	1310700.0	
bs_000	726	0.99	80360.55	84512.76	0	17300.00	50540.00	118635.00	1037240.0	
bt_000	167	1.00	59416.50	145446.44	0	862.83	30839.86	48787.93	2746564.8	
bu_000	691	0.99	4515324.70	10859903.84	0	105444.00	2359656.00	3863322.00	192871534.0	
bv_000	691	0.99	4515325.29	10859904.50	0	105444.00	2359656.00	3863322.00	192871534.0	
bx_000	3257	0.95	4112218.10	10351538.50	172	89649.00	2258824.00	3645960.00	186353854.0	
by_000	473	0.99	22028.93	53992.82	0	216.00	12628.00	20347.50	1002003.0	
bz_000	2723	0.95	101960.84	628912.90	0	6.00	1036.00	13674.00	40542588.0	
ca_000	4356	0.93	39168.82	36748.30	0	6886.00	25436.00	68004.50	120956.0	
cb_000	726	0.99	405638.15	369386.80	0	77125.00	278990.00	704580.00	1209520.0	
cc_000	3255	0.95	3803443.56	9625672.25	0	62416.00	2108912.00	3364634.00	148615188.0	
cd_000	676	0.99	1209600.00	0.00	1209600	1209600.00	1209600.00	1209600.00	1209600.0	
ce_000	2502	0.96	64343.56	142846.94	0	266.00	3409.00	87235.50	4908098.0	
cf_000	14862	0.75	47.07	1584.25	0	0.00	2.00	2.00	223146.0	
cg_000	14861	0.75	91.52	371.70	0	8.00	46.00	104.00	21400.0	
ch_000	14861	0.75	0.00	0.03	0	0.00	0.00	0.00	2.0	
ci_000	338	0.99	3481204.05	8355996.96	0	48249.84	1858641.12	2947265.52	140986129.9	
cj_000	338	0.99	102841.85	1135173.87	0	0.00	0.00	0.00	60949671.4	
ck_000	338	0.99	714342.70	2180714.30	0	14586.72	250267.20	549352.32	55428669.1	
cl_000	9553	0.84	343.02	4706.74	0	0.00	0.00	2.00	130560.0	
cm_000	9877	0.84	343.10	1693.36	0	0.00	8.00	100.00	73370.0	
cn_000	687	0.99	2336.67	61183.83	0	0.00	0.00	0.00	6278490.0	
cn_001	687	0.99	21951.49	245799.37	0	0.00	0.00	0.00	14512994.0	
cn_002	687	0.99	161050.88	1073892.81	0	0.00	0.00	7936.00	58508606.0	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
cn_003	687	0.99	531477.96	2216301.04	0	4622.00	34994.00	232066.00	94979324.0	
cn_004	687	0.99	1282835.14	3357254.34	0	19114.00	518462.00	1207694.00	169869316.0	
cn_005	687	0.99	1341058.71	3144659.17	0	5028.00	703524.00	1519808.00	117815764.0	
cn_006	687	0.99	410564.09	1294208.06	0	626.00	96266.00	449060.00	72080406.0	
cn_007	687	0.99	64425.13	406454.06	0	62.00	9976.00	31074.00	33143734.0	
cn_008	687	0.99	19226.79	186269.73	0	0.00	1852.00	5286.00	7541716.0	
cn_009	687	0.99	7820.47	266492.78	0	0.00	24.00	294.00	36398374.0	
co_000	14862	0.75	340.83	3204.48	0	0.00	8.00	72.00	317378.0	
cp_000	2724	0.95	570.40	7773.26	0	4.00	14.00	82.00	496360.0	
cq_000	691	0.99	4515325.18	10859904.09	0	105444.00	2359656.00	3863322.00	192871534.0	
cr_000	46329	0.23	37.06	1027.00	0	0.00	0.00	0.00	57450.0	
cs_000	669	0.99	5479.86	10313.98	0	1232.00	3192.00	5686.00	839240.0	
cs_001	669	0.99	788.43	2886.33	0	32.00	360.00	692.00	438806.0	
cs_002	669	0.99	238810.64	1215538.09	0	222.00	20570.00	94924.00	46085940.0	
cs_003	669	0.99	355373.12	1110963.71	0	3079.00	121780.00	295909.00	42421854.0	
cs_004	669	0.99	444228.28	2073936.55	0	2726.00	91080.00	208500.00	74860628.0	
cs_005	669	0.99	2235386.66	5613544.60	0	19186.00	1220860.00	2049318.00	379142116.0	
cs_006	669	0.99	545774.18	1168199.16	0	13360.00	240744.00	686260.00	73741974.0	
cs_007	669	0.99	14771.42	80051.22	0	1204.00	6104.00	18164.00	12884218.0	
cs_008	669	0.99	211.75	10153.37	0	2.00	46.00	148.00	1584558.0	
cs_009	669	0.99	779.20	184358.48	0	0.00	0.00	0.00	44902992.0	
ct_000	13808	0.77	749.09	5585.95	0	40.00	210.00	672.00	910366.0	
cu_000	13808	0.77	1222.96	7633.44	0	82.00	278.00	856.00	733688.0	
cv_000	13808	0.77	1928824.89	3660900.68	0	23898.50	1181117.00	2400717.00	81610510.0	
cx_000	13808	0.77	351510.24	1512775.91	0	944.00	44465.00	126794.00	44105494.0	
cy_000	13808	0.77	274.18	8804.14	0	0.00	0.00	0.00	931472.0	
cz_000	13808	0.77	19374.29	247184.31	0	4.00	202.00	6343.00	19156530.0	
da_000	13808	0.77	7.39	190.49	0	0.00	0.00	0.00	21006.0	
db_000	13808	0.77	13.42	73.41	0	0.00	0.00	18.00	9636.0	
dc_000	13808	0.77	2200752.21	4110149.89	0	26558.50	1734472.00	2644937.50	120759484.0	
dd_000	2503	0.96	3123.96	9516.68	0	132.00	1354.00	2678.00	445142.0	
de_000	2724	0.95	375.15	1689.06	0	66.00	144.00	296.00	176176.0	
df_000	4008	0.93	2718.64	137333.13	0	0.00	0.00	0.00	21613910.0	
dg_000	4008	0.93	5609.96	208564.92	0	0.00	0.00	0.00	27064294.0	
dh_000	4008	0.93	4707.07	560279.86	0	0.00	0.00	0.00	124700880.0	
di_000	4006	0.93	37248.24	425438.17	0	0.00	0.00	0.00	22987424.0	
dj_000	4007	0.93	39.94	4533.14	0	0.00	0.00	0.00	726750.0	
dk_000	4007	0.93	1861.31	66597.86	0	0.00	0.00	0.00	5483574.0	
dl_000	4008	0.93	28541.77	1095662.13	0	0.00	0.00	0.00	103858120.0	
dm_000	4009	0.93	7923.23	277535.57	0	0.00	0.00	0.00	23697916.0	
dn_000	691	0.99	33745.45	97337.19	0	660.00	14330.00	27340.00	2924584.0	
do_000	2724	0.95	28507.85	61254.76	0	20.00	10377.00	37672.00	1874542.0	
dp_000	2726	0.95	6958.65	13955.45	0	6.00	2532.00	8319.50	348118.0	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
dq_000	2767	0.95	2202368.89	37130587.48	0	0.00	0.00	0.00	2021871166.0	
dr_000	2726	0.95	203759.85	1366064.70	0	0.00	0.00	0.00	50137662.0	
ds_000	2727	0.95	89655.00	208201.63	0	684.00	47940.00	99202.00	4970962.0	
dt_000	2727	0.95	15403.35	33801.02	0	150.00	8316.00	17630.00	656432.0	
du_000	2726	0.95	4058711.62	11567771.48	0	5380.00	185400.00	3472540.00	460207620.0	
dv_000	2726	0.95	593834.98	2082997.65	0	742.00	30592.00	534655.50	127034534.0	
dx_000	2723	0.95	791208.48	4151033.42	0	0.00	0.00	8764.00	114288420.0	
dy_000	2724	0.95	7780.35	59244.49	0	0.00	0.00	36.00	3793022.0	
dz_000	2723	0.95	0.22	10.82	0	0.00	0.00	0.00	1414.0	
ea_000	2723	0.95	1.57	53.53	0	0.00	0.00	0.00	8506.0	
eb_000	4007	0.93	9717093.23	42746746.40	0	0.00	622110.00	4000270.00	1322456920.0	
ec_00	10239	0.83	1353.13	3536.64	0	114.76	754.40	1379.38	106020.2	
ed_000	9553	0.84	1452.16	3525.03	0	98.00	832.00	1504.00	82806.0	
ee_000	671	0.99	733404.21	2416165.66	0	15698.00	260704.00	573060.00	74984446.0	
ee_001	671	0.99	783874.59	2570110.88	0	8536.00	346940.00	667390.00	98224378.0	
ee_002	671	0.99	445489.73	1155539.82	0	2936.00	233796.00	438396.00	77933926.0	
ee_003	671	0.99	211126.45	543318.82	0	1166.00	112086.00	218232.00	37758390.0	
ee_004	671	0.99	445734.31	1168313.93	0	2700.00	221518.00	466614.00	97152378.0	
ee_005	671	0.99	393946.20	1121044.41	0	3584.00	189988.00	403222.00	57435236.0	
ee_006	671	0.99	333058.24	1069159.70	0	512.00	92432.00	275094.00	31607814.0	
ee_007	671	0.99	346271.43	1728056.01	0	110.00	41098.00	167814.00	119580108.0	
ee_008	671	0.99	138729.98	449510.05	0	0.00	3812.00	139724.00	19267396.0	
ee_009	671	0.99	8388.91	47470.43	0	0.00	0.00	2028.00	3810078.0	
ef_000	2724	0.95	0.09	4.37	0	0.00	0.00	0.00	482.0	
eg_000	2723	0.95	0.21	8.83	0	0.00	0.00	0.00	1146.0	

4 Limpieza de los datos

Se puede apreciar cómo muchos de los atributos tienes ceros hasta el tercer cuartil, esto es algo que se debe revisar para entender a qué se debe y cómo gestionarlo. También hay valores atributos que presentan una baja completitud, para tratar los datos nulos (missing values) comenzamos por visualizar el porcentaje de valores perdidos de cada atributo; los mostramos en orden descendente para poder ver claramente qué atributos contienen mayor porcentaje de valores perdidos.

4.1 Gestión de ceros y datos perdidos

```
# Estadísticas de valores vacíos
sort(colMeans(is.na(rtrain)), decreasing = TRUE)
```

```
##      br_000      bq_000      bp_000      bo_000      ab_000      cr_000
## 0.821066667 0.812033333 0.795666667 0.772216667 0.772150000 0.772150000
##      bn_000      bm_000      bl_000      bk_000      ad_000      cf_000
## 0.733483333 0.659150000 0.454616667 0.383900000 0.247700000 0.247700000
##      co_000      cg_000      ch_000      ct_000      cu_000      cv_000
## 0.247700000 0.247683333 0.247683333 0.230133333 0.230133333 0.230133333
##      cx_000      cy_000      cz_000      da_000      db_000      dc_000
## 0.230133333 0.230133333 0.230133333 0.230133333 0.230133333 0.230133333
##       ec_00      cm_000      cl_000      ed_000      ak_000      ca_000
## 0.170650000 0.164616667 0.159216667 0.159216667 0.073333333 0.072600000
##      dm_000      df_000      dg_000      dh_000      dl_000      dj_000
## 0.066816667 0.066800000 0.066800000 0.066800000 0.066800000 0.066783333
##      dk_000      eb_000      di_000      ac_000      bx_000      cc_000
## 0.066783333 0.066783333 0.066766667 0.055583333 0.054283333 0.054250000
##      dq_000      bd_000      ds_000      dt_000      dp_000      dr_000
## 0.046116667 0.045450000 0.045450000 0.045450000 0.045433333 0.045433333
##      du_000      dv_000      bc_000      cp_000      de_000      do_000
## 0.045433333 0.045433333 0.045416667 0.045400000 0.045400000 0.045400000
##      dy_000      ef_000      ar_000      bz_000      dx_000      dz_000
## 0.045400000 0.045400000 0.045383333 0.045383333 0.045383333 0.045383333
##      ea_000      eg_000      be_000      dd_000      ce_000      ax_000
## 0.045383333 0.045383333 0.041716667 0.041716667 0.041700000 0.041683333
##      ae_000      af_000      av_000      bf_000      bs_000      cb_000
## 0.041666667 0.041666667 0.041666667 0.041666667 0.012100000 0.012100000
##      bu_000      bv_000      cq_000      dn_000      ba_000      ba_001
## 0.011516667 0.011516667 0.011516667 0.011516667 0.011466667 0.011466667
##      ba_002      ba_003      ba_004      ba_005      ba_006      ba_007
## 0.011466667 0.011466667 0.011466667 0.011466667 0.011466667 0.011466667
##      ba_008      ba_009      cn_000      cn_001      cn_002      cn_003
## 0.011466667 0.011466667 0.011450000 0.011450000 0.011450000 0.011450000
##      cn_004      cn_005      cn_006      cn_007      cn_008      cn_009
## 0.011450000 0.011450000 0.011450000 0.011450000 0.011450000 0.011450000
##      cd_000      ag_000      ag_001      ag_002      ag_003      ag_004
## 0.011266667 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333
##      ag_005      ag_006      ag_007      ag_008      ag_009      ay_000
## 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333
##      ay_001      ay_002      ay_003      ay_004      ay_005      ay_006
## 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333
##      ay_007      ay_008      ay_009      az_000      az_001      az_002
## 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333
##      az_003      az_004      az_005      az_006      az_007      az_008
## 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333
##      az_009      ee_000      ee_001      ee_002      ee_003      ee_004
## 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333
##      ee_005      ee_006      ee_007      ee_008      ee_009      cs_000
## 0.011183333 0.011183333 0.011183333 0.011183333 0.011183333 0.011150000
##      cs_001      cs_002      cs_003      cs_004      cs_005      cs_006
## 0.011150000 0.011150000 0.011150000 0.011150000 0.011150000 0.011150000
##      cs_007      cs_008      cs_009      ah_000      bb_000      al_000
## 0.011150000 0.011150000 0.011150000 0.010750000 0.010750000 0.010700000
##      an_000      ap_000      bg_000      bh_000      ai_000      aj_000
## 0.010700000 0.010700000 0.010700000 0.010700000 0.010483333 0.010483333
##        am_0      as_000      at_000      au_000      ao_000      aq_000
## 0.010483333 0.010483333 0.010483333 0.010483333 0.009816667 0.009816667
##      bi_000      bj_000      by_000      ci_000      cj_000      ck_000
## 0.009816667 0.009816667 0.007883333 0.005633333 0.005633333 0.005633333
##      bt_000       class      aa_000
## 0.002783333 0.000000000 0.000000000
```

Se puede ver que hay atributos con más de un 20% de valores perdidos, por lo tanto se ha decicido eliminar estos atributos y trabajar únicamente con el resto.De hecho, existen 8 variables con más del 50% de los valores perdidos. Tras eliminar las variables con más del 20% de los valores perdidos nos quedamos con 147 variables.

```
columns_to_remove <- which(colMeans(is.na(rtrain)) > 0.2)
ctrain = subset(rtrain, select = -c(columns_to_remove) )
ctest = subset(rtest, select = -c(columns_to_remove) )
dim(ctrain)
```

```
## [1] 60000   147
```

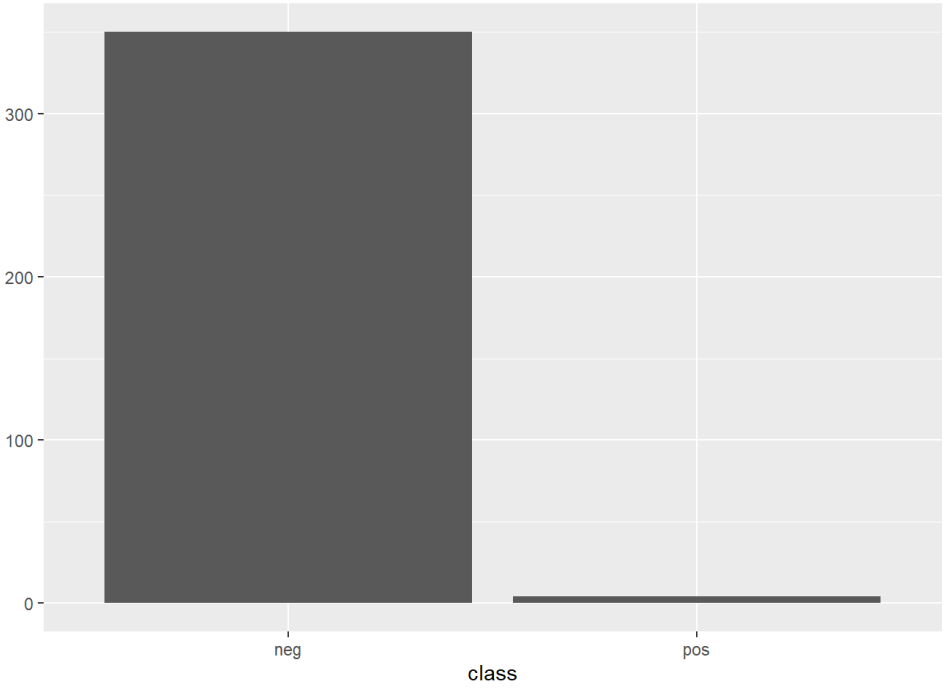
El resto de valores perdidos deberíamos imputarlos o eliminarlos, pero antes hay una duda que resolver, igual que hay columnas con un alto porcentaje de valores perdidos, es posible que también haya filas con un alto porcentaje de valores perdidos. Alrededor de 180 filas contienen más del 95% de valores perdidos y 354 filas contienen más del 50% de sus valores perdidos.

```
head(sort(rowMeans(is.na(ctrain)), decreasing = TRUE),200)
```

```
##   [1] 0.9863946 0.9863946 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##   [8] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [15] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [22] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [29] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [36] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [43] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [50] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [57] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [64] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [71] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [78] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [85] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [92] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
##  [99] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## [106] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## [113] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## [120] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## [127] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## [134] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## [141] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## [148] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## [155] 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918 0.9795918
## [162] 0.9727891 0.9727891 0.9727891 0.9727891 0.9727891 0.9727891 0.9727891
## [169] 0.9727891 0.9727891 0.9727891 0.9659864 0.9659864 0.9659864 0.9659864
## [176] 0.9659864 0.9659864 0.9659864 0.9659864 0.9659864 0.9659864 0.9659864
## [183] 0.9659864 0.9591837 0.9591837 0.9319728 0.9183673 0.8639456 0.8163265
## [190] 0.8095238 0.8095238 0.8095238 0.8095238 0.8095238 0.8095238 0.8027211
## [197] 0.7959184 0.7959184 0.7959184 0.7959184
```

Podemos ver que hay filas en las que el porcentaje de nulos es superior al 50%, es decir que desconocemos la mayoría de valores de esta fila. Veamos si el hecho de desconocer estos valores influye en que la clase sea negativa o positiva.

```
null_rows <- filter(ctrain, rowMeans(is.na(ctrain)) > 0.5)
qplot(as.factor(null_rows$class), xlab = "class")
```




```
prop.table(table(null_rows$class))
```

```
##
##        neg        pos
## 0.98870056 0.01129944
```

Podemos ver cómo la distribución obtenida para estos datos es muy parecida a la del total de los datos, por lo tanto, el hecho de desconocer la mayoría de valores de una fila no influye en que la clase sea negativa o positiva. Esto permite eliminar estos registros de nuestro data set.

```
ctrain <- filter(ctrain, rowMeans(is.na(ctrain)) < 0.5)
dim(ctrain)
```

```
## [1] 59646   147
```

Existen varios métodos para imputar los valores perdidos. Una opción podría ser eliminar los valores perdidos. Otra opción sería sustituir estos valores perdidos por un estadístico de la variable, como puede ser la media, mediana o moda. Este enfoque puede dar buenos resultados cuando la variabilidad de la variable es pequeña. Otros métodos más ajustados optan por una imputación que tenga en cuenta múltiples valores, como pueden ser km basado en la búsqueda de vecinos cercanos o árboles de decisión (missforest).

En este momento se van a realizar dos tipos de imputaciones, por la mediana y mediante vecinos (paquete ClustImpute)[[4]] paginaweb4 (https://www.r-bloggers.com/2019/06/intoducing-clustimpute-a-new-approach-for-k-means-clustering-with-build-in-missing-data-imputation/), cada uno de ellos con un nombre de conjunto de datos tratado diferente. En la parte de análisis se comparará los resultados obtenidos mediante cada tipo de imputación. El tiempo de ejecución entre los dos tipos de imputación es significativamente diferente, casi instantáneo en el caso de imputación por la mediana y más de 2 minutos para el caso de k-Means.

Debe comentarse en este apartado que **no se ha podido utilizar el algoritmo de imputación missForest**. Los autores de este trabajo suponen que se debe al alto coste computacional de implementación de esta técnica [5] (https://rpubs.com/lmorgan95/MissForest) sobre un conjunto de datos tan grande (training original de 60000 filas x 171 columnas). El hecho es que R se quedaba bloqueado al intentar aplicar la técnica, por lo que se ha tenido que desistir en su uso.

La ventaja del algoritmo de ClustImpute es que es computacionalmente más eficiente que otros, como el paquete MICE o missForest, los cuales no han sido posibles implantar con el total de los casos.

```
# Imputación mediana
library(imputeMissings)
```

```
##
## Attaching package: 'imputeMissings'
```

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

```
ctrain_median <- impute(ctrain, object = NULL, method = "median/mode", flag = FALSE)
ctest_median <- impute(ctest, object = NULL, method = "median/mode", flag = FALSE)
```

```
# Imputación k-means
library(ClustImpute)
## k-means a training
ctrain_num <- ctrain[,2:147]
t0<-proc.time()
train.km <- ClustImpute(ctrain_num, nr_cluster=7)
t1<-proc.time()
t1-t0
```

```
##    user  system elapsed
##  155.23    6.34  164.41
```

```
ctrain_km<-ctrain[,1]
ctrain_km <- cbind(ctrain_km,train.km$complete_data)
eda_skim<-skim(ctrain_km)
eda_skim
```

Data summary

Name	ctrain_km
Number of rows	59646

Number of columns	147
_____	
Column type frequency:	
character	1
numeric	146
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
class	0	1	3	3	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
aa_000	0	1	59249.58	145429.37	0	840.00	30801.00	48566.00	2746564.0	▇▁▁▁▁
ac_000	0	1	338864375.61	779230895.57	0	16.00	156.00	966.00	2130706796.0	▇▁▁▁▁
ae_000	0	1	7.73	186.55	0	0.00	0.00	0.00	21050.0	▇▁▁▁▁
af_000	0	1	12.87	230.18	0	0.00	0.00	0.00	20070.0	▇▁▁▁▁
ag_000	0	1	221.18	20424.39	0	0.00	0.00	0.00	3376892.0	▇▁▁▁▁
ag_001	0	1	966.14	34090.37	0	0.00	0.00	0.00	4109372.0	▇▁▁▁▁
ag_002	0	1	8525.08	149658.07	0	0.00	0.00	0.00	10552856.0	▇▁▁▁▁
ag_003	0	1	88022.68	759045.70	0	0.00	0.00	0.00	63402074.0	▇▁▁▁▁
ag_004	0	1	436019.58	2377957.13	0	308.00	3650.00	49340.00	228830570.0	▇▁▁▁▁
ag_005	0	1	1104713.30	3254157.20	0	13733.00	175214.00	912128.00	179187978.0	▇▁▁▁▁
ag_006	0	1	1653630.22	3900888.61	0	10537.00	924061.00	1883719.50	94020666.0	▇▁▁▁▁
ag_007	0	1	497989.28	1419563.74	0	0.00	118181.00	587412.00	63346754.0	▇▁▁▁▁
ag_008	0	1	35450.83	219545.65	0	0.00	1758.00	26580.50	17702522.0	▇▁▁▁▁
ag_009	0	1	5095.72	169207.91	0	0.00	0.00	362.00	25198514.0	▇▁▁▁▁
ah_000	0	1	1818044.44	4198436.62	0	29737.50	1003197.00	1604922.50	74247318.0	▇▁▁▁▁
ai_000	0	1	9208.14	167707.86	0	0.00	0.00	0.00	16512852.0	▇▁▁▁▁
aj_000	0	1	1140.24	50245.32	0	0.00	0.00	0.00	5629340.0	▇▁▁▁▁
ak_000	0	1	1700.32	106858.64	0	0.00	0.00	0.00	10444924.0	▇▁▁▁▁
al_000	0	1	59635.96	544888.99	0	0.00	0.00	1202.00	34762578.0	▇▁▁▁▁
am_0	0	1	94124.73	863761.25	0	0.00	0.00	2366.00	55903508.0	▇▁▁▁▁
an_000	0	1	3475208.00	7810469.60	0	73291.50	1920404.00	3130268.50	140861830.0	▇▁▁▁▁
ao_000	0	1	3009440.90	6827873.39	0	65499.50	1644440.00	2675962.00	122201822.0	▇▁▁▁▁
ap_000	0	1	1008384.05	3093260.83	0	25126.50	357533.00	725717.50	77934944.0	▇▁▁▁▁
aq_000	0	1	443195.02	1263670.13	0	4152.00	178778.00	377236.00	25562646.0	▇▁▁▁▁
ar_000	0	1	0.62	6.36	0	0.00	0.00	0.00	350.0	▇▁▁▁▁
as_000	0	1	124.25	10974.78	0	0.00	0.00	0.00	1655240.0	▇▁▁▁▁
at_000	0	1	5092.73	119501.26	0	0.00	0.00	0.00	10400504.0	▇▁▁▁▁
au_000	0	1	233.72	15795.99	0	0.00	0.00	0.00	2626676.0	▇▁▁▁▁
av_000	0	1	1226.83	6943.91	0	10.00	120.00	678.00	794458.0	▇▁▁▁▁
ax_000	0	1	413.35	1706.18	0	10.00	68.00	272.00	116652.0	▇▁▁▁▁
ay_000	0	1	12146.75	453287.81	0	0.00	0.00	0.00	50553892.0	▇▁▁▁▁

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ay_001	0	1	10211.52	534157.71	0	0.00	0.00	0.00	80525378.0	
ay_002	0	1	11354.40	440309.32	0	0.00	0.00	0.00	28474838.0	
ay_003	0	1	7190.31	205919.54	0	0.00	0.00	0.00	13945170.0	
ay_004	0	1	10514.59	353684.22	0	0.00	0.00	0.00	40028704.0	
ay_005	0	1	111788.87	1392397.04	0	0.00	0.00	39943.00	124948914.0	
ay_006	0	1	1076093.54	3271492.73	0	0.00	166941.00	1267666.50	127680326.0	
ay_007	0	1	1540736.52	5093964.00	0	6067.00	344905.00	1334642.00	489678156.0	
ay_008	0	1	1047667.20	3981525.83	0	7500.00	94344.00	610217.00	104566992.0	
ay_009	0	1	1156.52	97700.17	0	0.00	0.00	0.00	18824656.0	
az_000	0	1	7826.54	73445.95	0	1028.00	2096.00	4144.00	10124620.0	
az_001	0	1	4433.24	34465.12	0	58.00	634.00	2011.50	4530258.0	
az_002	0	1	8047.90	106350.82	0	88.00	1014.00	3130.00	14217662.0	
az_003	0	1	87094.53	651626.10	0	290.00	3556.00	42962.00	45584242.0	
az_004	0	1	1473705.48	4175992.81	0	1534.00	81156.00	1771420.00	123047106.0	
az_005	0	1	2128845.36	6447336.57	0	38290.00	524415.00	1790294.50	467832334.0	
az_006	0	1	101690.82	898744.09	0	10.00	290.00	4217.00	64589140.0	
az_007	0	1	17346.67	279984.76	0	0.00	0.00	0.00	39158218.0	
az_008	0	1	658.95	14885.23	0	0.00	0.00	0.00	1947884.0	
az_009	0	1	41.99	3248.08	0	0.00	0.00	0.00	666148.0	
ba_000	0	1	1395496.30	3768897.34	0	33364.50	677502.00	1274183.50	232871714.0	
ba_001	0	1	891557.25	2340619.46	0	14754.00	441958.00	809820.00	116283282.0	
ba_002	0	1	411486.78	1191952.10	0	5170.00	185496.00	339472.00	55807388.0	
ba_003	0	1	273160.75	746684.63	0	1816.00	133687.00	243967.50	36931418.0	
ba_004	0	1	204298.88	538163.99	0	608.00	101646.00	196734.50	25158556.0	
ba_005	0	1	188449.14	508621.90	0	372.00	83728.00	184264.50	19208664.0	
ba_006	0	1	210040.46	636181.30	0	350.00	69453.00	204402.00	18997660.0	
ba_007	0	1	185205.79	523915.60	0	72.00	4320.00	206871.50	14314086.0	
ba_008	0	1	35738.49	243022.31	0	0.00	22.00	1798.00	31265984.0	
ba_009	0	1	35691.19	334851.35	0	0.00	0.00	60.00	43706408.0	
bb_000	0	1	4550995.42	10931314.08	0	105722.50	2362828.00	3871244.00	192871534.0	
bc_000	0	1	701.70	4449.02	0	0.00	16.00	152.00	396952.0	
bd_000	0	1	1107.40	5632.18	0	10.00	72.00	472.00	306452.0	
be_000	0	1	1659.74	10722.24	0	20.00	190.00	654.00	810568.0	
bf_000	0	1	91.29	621.70	0	0.00	2.00	20.00	51050.0	
bg_000	0	1	1815348.11	4190613.19	0	29746.00	1002979.00	1604485.50	74247318.0	
bh_000	0	1	58196.83	152674.80	0	852.00	26383.00	49120.00	3200582.0	
bi_000	0	1	493713.75	1486843.79	0	15872.50	179756.00	379920.00	44937496.0	
bj_000	0	1	511944.99	1823790.20	0	8516.00	154609.00	334049.50	45736316.0	
bs_000	0	1	80421.24	84606.74	0	17300.00	50540.00	118680.00	1037240.0	
bt_000	0	1	59346.51	145515.28	0	863.00	30854.50	48667.35	2746564.8	
bu_000	0	1	4547906.13	10926130.41	0	105550.50	2361778.00	3868656.00	192871534.0	
bv_000	0	1	4550321.36	10934833.22	0	105722.50	2362835.00	3871151.50	192871534.0	
bx_000	0	1	4294494.41	10768216.94	172	91997.00	2309897.00	3711940.50	186353854.0	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
by_000	0	1	21989.87	53915.21	0	217.00	12627.00	20333.00	1002003.0	
bz_000	0	1	126202.40	779628.79	0	10.00	1158.00	15403.50	40542588.0	
ca_000	0	1	39825.79	36869.15	0	7204.00	26629.00	68972.00	120956.0	
cb_000	0	1	405568.68	369413.26	0	77160.00	278590.00	704460.00	1209520.0	
cc_000	0	1	3980334.70	10048374.49	0	65437.00	2153239.00	3430212.50	148615188.0	
cd_000	0	1	1209600.00	0.00	1209600	1209600.00	1209600.00	1209600.00	1209600.0	
ce_000	0	1	70124.35	166827.30	0	248.00	3480.00	88641.00	4908098.0	
ci_000	0	1	3479382.75	8349369.36	0	48375.84	1860365.28	2945650.56	140986129.9	
cj_000	0	1	103208.89	1144579.52	0	0.00	0.00	0.00	60949671.4	
ck_000	0	1	713929.84	2180076.93	0	14626.08	250299.84	549352.32	55428669.1	
cl_000	0	1	421.21	5242.19	0	0.00	0.00	2.00	130560.0	
cm_000	0	1	406.28	2022.59	0	0.00	6.00	98.00	73370.0	
cn_000	0	1	2318.81	60970.36	0	0.00	0.00	0.00	6278490.0	
cn_001	0	1	21795.88	244738.17	0	0.00	0.00	0.00	14512994.0	
cn_002	0	1	160125.86	1069410.40	0	0.00	0.00	7940.00	58508606.0	
cn_003	0	1	528921.96	2207309.44	0	4606.00	34750.00	231023.50	94979324.0	
cn_004	0	1	1278759.49	3348640.70	0	18932.50	516146.00	1205614.00	169869316.0	
cn_005	0	1	1337095.20	3137707.65	0	4952.50	698720.00	1517969.50	117815764.0	
cn_006	0	1	409562.72	1291202.56	0	624.00	95495.00	448396.50	72080406.0	
cn_007	0	1	64231.85	405443.48	0	60.00	9932.00	31013.50	33143734.0	
cn_008	0	1	19137.80	185754.27	0	0.00	1840.00	5274.00	7541716.0	
cn_009	0	1	7783.02	265750.00	0	0.00	24.00	294.00	36398374.0	
cp_000	0	1	810.70	11132.38	0	4.00	14.00	86.00	496360.0	
cq_000	0	1	4554087.20	10954363.90	0	105550.50	2362563.00	3868418.00	192871534.0	
cs_000	0	1	5468.66	10289.66	0	1232.00	3186.00	5676.00	839240.0	
cs_001	0	1	786.66	2880.75	0	32.00	360.00	692.00	438806.0	
cs_002	0	1	237908.24	1212463.09	0	220.00	20489.00	94590.50	46085940.0	
cs_003	0	1	354191.52	1108435.19	0	3050.50	121215.00	295115.00	42421854.0	
cs_004	0	1	442722.08	2067562.63	0	2714.00	90862.00	208148.00	74860628.0	
cs_005	0	1	2228960.97	5600699.64	0	19030.50	1216469.00	2046225.00	379142116.0	
cs_006	0	1	544342.14	1159573.96	0	13246.00	239516.00	685365.00	73741974.0	
cs_007	0	1	14728.70	79741.38	0	1200.00	6086.00	18144.00	12884218.0	
cs_008	0	1	211.08	10126.53	0	2.00	46.00	148.00	1584558.0	
cs_009	0	1	775.08	183871.02	0	0.00	0.00	0.00	44902992.0	
dd_000	0	1	3680.24	11356.70	0	136.00	1416.00	2810.00	445142.0	
de_000	0	1	426.05	1947.88	0	68.00	148.00	308.00	176176.0	
df_000	0	1	4266.92	171919.03	0	0.00	0.00	0.00	21613910.0	
dg_000	0	1	10472.81	309404.51	0	0.00	0.00	0.00	27064294.0	
dh_000	0	1	5241.25	557443.96	0	0.00	0.00	0.00	124700880.0	
di_000	0	1	56558.62	593424.85	0	0.00	0.00	0.00	22987424.0	
dj_000	0	1	51.13	5314.50	0	0.00	0.00	0.00	726750.0	
dk_000	0	1	2620.68	87615.10	0	0.00	0.00	0.00	5483574.0	
dl_000	0	1	28877.70	1086812.07	0	0.00	0.00	0.00	103858120.0	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
dm_000	0	1	8103.92	288339.56	0	0.00	0.00	0.00	23697916.0	
dn_000	0	1	34031.72	97996.76	0	660.00	14356.00	27378.00	2924584.0	
do_000	0	1	31568.79	71819.14	0	24.00	12421.00	38866.00	1874542.0	
dp_000	0	1	7648.97	16154.80	0	6.00	2923.00	8636.00	348118.0	
dq_000	0	1	4478399.28	65690174.92	0	0.00	0.00	0.00	2021871166.0	
dr_000	0	1	274100.09	1676593.50	0	0.00	0.00	0.00	50137662.0	
ds_000	0	1	105612.77	256855.11	0	738.00	51340.00	103384.00	4970962.0	
dt_000	0	1	17921.65	40947.85	0	160.00	8909.00	18366.00	656432.0	
du_000	0	1	4499672.03	13268482.90	0	5820.00	199880.00	3671100.00	460207620.0	
dv_000	0	1	707231.47	2792677.36	0	804.00	33682.00	565707.00	127034534.0	
dx_000	0	1	1036120.26	5258684.62	0	0.00	0.00	9729.00	114288420.0	
dy_000	0	1	10036.40	71038.32	0	0.00	0.00	40.00	3793022.0	
dz_000	0	1	0.27	12.31	0	0.00	0.00	0.00	1414.0	
ea_000	0	1	1.62	53.03	0	0.00	0.00	0.00	8506.0	
eb_000	0	1	10431341.35	43496326.69	0	0.00	716365.00	4728405.00	1322456920.0	
ec_00	0	1	1569.56	4268.89	0	120.03	783.16	1426.99	106020.2	
ed_000	0	1	1596.67	3965.20	0	98.00	860.00	1546.00	82806.0	
ee_000	0	1	731551.55	2412145.51	0	15647.00	259915.00	571167.00	74984446.0	
ee_001	0	1	781834.38	2564280.61	0	8520.00	346183.00	665155.50	98224378.0	
ee_002	0	1	444275.31	1153227.65	0	2918.00	232971.00	437626.00	77933926.0	
ee_003	0	1	210605.19	542355.24	0	1158.00	111498.00	217821.00	37758390.0	
ee_004	0	1	444859.88	1166455.39	0	2676.00	220665.00	465964.00	97152378.0	
ee_005	0	1	393217.71	1118875.25	0	3560.00	189658.00	402713.00	57435236.0	
ee_006	0	1	332024.50	1066556.90	0	506.00	91785.00	274440.00	31607814.0	
ee_007	0	1	345109.26	1724996.84	0	108.00	40682.00	167518.00	119580108.0	
ee_008	0	1	138338.85	448484.21	0	0.00	3759.00	139257.50	19267396.0	
ee_009	0	1	8362.30	47357.06	0	0.00	0.00	2010.00	3810078.0	
ef_000	0	1	0.11	4.83	0	0.00	0.00	0.00	482.0	
eg_000	0	1	0.26	9.59	0	0.00	0.00	0.00	1146.0	

```
## km a testing
ctest_num <- ctest[,2:147]
test.km <- ClustImpute(ctest_num, nr_cluster=5)
ctest_km<-ctest[,1]
ctest_km <- cbind(ctest_km,test.km$complete_data)
```

Se realiza una muestra reducida (1992 casos) de la muestra de training (996 de clase negativa y 996 (todos) de la positiva) para ejecutar missForest. El tiempo de ejecución ha sido 33,8 min. Extrapolando, podría estimarse que (de ser satisfactorio) se necesitarían más de 21 horas para la imputación de 128 variables sobre un conjunto de aproximadadamente 76000 filas (60000 de training y 16000 filas), menos los eliminados por muy elevado porcentaje de perdidos. Se comentan las líneas de la implementación de missForest sobre la muestra, por el alto tiempo de ejecución.

```
# table(ctrain$class)
# ctrain_neg<-ctrain[ctrain$class=="neg"]
# ctrain_pos<-ctrain[ctrain$class=="pos"]
# indices_neg <- sample( 1:nrow(ctrain_neg), 996 )
# indices_pos <- sample( 1:nrow(ctrain_pos), 996 )
# ctrain_sample<-ctrain_neg[indices_neg,]
# ctrain_sample<-rbind(ctrain_sample,ctrain_pos[indices_pos,])
# ctrain_neg<-NULL
# ctrain_pos<-NULL
#
# library(missForest)
# eda_skim<-skim(ctrain_sample)
# eda_skim
# dim(ctrain_sample)
# t2<-proc.time()
# ctrains_missForest<- missForest(ctrain_sample[1:1992,-1])
# ctrains_mF<-ctrains_missForest$ximp
# t3<-proc.time()
# t3-t2
```

Una vez tratados los valores perdidos, se procede al tratamiento de ceros, sobre el conjunto de datos con imputación por vecindad. Como se ha visto antes hay atributos en los que hay ceros hasta en el tercer cuartil, vamos a investigar a qué se debe. En la descripción del dataset, se indica que hay 7 variables que están binarizadas en 10 partes, es decir, que hay atributos que ocupan 10 columnas. Seleccionamos los atributos binarizados.

```
histogram <- ctrain_km[,c(6:15, 32:41, 42:51, 52:61, 88:97, 100:109, 136:145)]
dim(histogram)
```

```
## [1] 59646    70
```

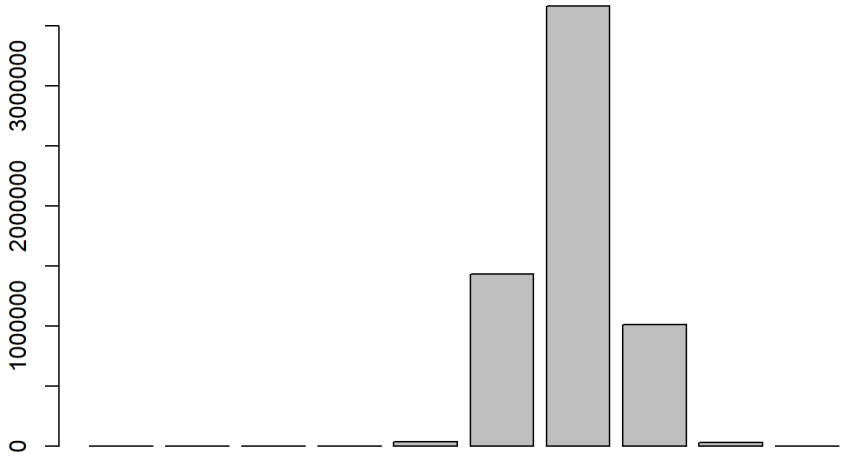
```
str(histogram)
```

```
## Classes 'data.table' and 'data.frame':   59646 obs. of  70 variables:
##  $ ag_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ag_001: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ag_002: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ag_003: int  0 0 0 318 0 0 0 0 0 11804 ...
##  $ ag_004: int  37250 18254 1648 2212 43752 9128 1202 2130 458 684444 ...
##  $ ag_005: int  1432864 653294 370592 3232 1966618 701702 3766 142462 440704 326536 ...
##  $ ag_006: int  3664156 1720800 1883374 1872 1800340 1462836 1150 4227340 4398806 31586 ...
##  $ ag_007: int  1007684 516724 292936 0 131646 449716 0 1674278 2179182 0 ...
##  $ ag_008: int  25896 31642 12016 0 4588 39000 0 59718 144418 0 ...
##  $ ag_009: int  0 0 0 0 0 660 0 876 808 0 ...
##  $ ay_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ay_001: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ay_002: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ay_003: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ay_004: int  0 0 0 0 42124 0 0 0 15198 0 ...
##  $ ay_005: int  469014 71510 0 0 372236 280112 0 34444 123920 0 ...
##  $ ay_006: int  4239660 772720 870456 0 2128914 1160742 0 1558518 898176 0 ...
##  $ ay_007: int  703300 1996924 239798 2038 819596 774914 0 3996204 1096132 94188 ...
##  $ ay_008: int  755876 99560 1450312 5596 584074 447274 6118 517638 5030950 960182 ...
##  $ ay_009: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ az_000: int  5374 7336 1620 64 1644 1580 1484 6634 5574 250 ...
##  $ az_001: int  2108 7808 1156 6 362 680 14 6406 1170 16 ...
##  $ az_002: int  4114 13776 1228 6 562 886 24 12848 2110 32 ...
##  $ az_003: int  12348 13086 34250 914 842 1270 16 34152 3372 146 ...
##  $ az_004: int  615248 1010074 1811606 76 30194 42450 88 427644 61402 126 ...
##  $ az_005: int  5526276 1873902 710672 2478 3911734 2615652 4492 5618652 7090548 951126 ...
##  $ az_006: int  2378 14726 34 2398 1606 524 0 468 174 91162 ...
##  $ az_007: int  4 6 0 1692 0 0 0 0 20 11512 ...
##  $ az_008: int  0 0 0 0 0 0 0 0 4 0 ...
##  $ az_009: int  0 0 0 0 0 0 0 0 2 0 ...
##  $ ba_000: int  2328746 1378576 790690 6176 1348578 1037372 4332 1699220 3982850 702994 ...
##  $ ba_001: int  1022304 447166 672026 340 1035668 710296 646 1550888 1580644 186172 ...
##  $ ba_002: int  415432 199512 332340 304 338762 244992 316 619780 567792 69282 ...
##  $ ba_003: int  287230 154298 254892 102 236540 145738 54 445704 293120 36640 ...
##  $ ba_004: int  310246 137280 189596 74 182278 99242 140 323882 231626 20698 ...
##  $ ba_005: int  681504 138668 135758 406 151778 70704 444 261558 372878 17720 ...
##  $ ba_006: int  1118814 165908 103552 216 163248 90386 122 337086 135466 18586 ...
##  $ ba_007: int  3574 229652 81666 16 470800 260950 16 868682 0 2278 ...
##  $ ba_008: int  0 87082 46 0 19292 3362 14 4 0 0 ...
##  $ ba_009: int  0 4708 0 0 0 0 34 0 0 0 ...
##  $ cn_000: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ cn_001: int  0 0 0 52 0 0 0 0 0 0 ...
##  $ cn_002: int  0 38 0 2544 356 366 520 0 0 32854 ...
##  $ cn_003: int  118196 98644 33276 1894 378910 171324 1144 25100 24898 113024 ...
##  $ cn_004: int  1309472 1179502 1215280 2170 2497104 978164 3912 1693440 1446554 413450 ...
##  $ cn_005: int  3247182 1286736 1102798 822 993000 1182246 422 3593600 3471774 341800 ...
##  $ cn_006: int  1381362 336388 196502 152 64230 302306 114 756974 1999718 110462 ...
##  $ cn_007: int  98822 36294 10260 0 10482 23856 6 29974 206518 40426 ...
##  $ cn_008: int  11208 5192 2422 0 2776 4580 0 7430 13860 2354 ...
##  $ cn_009: int  1608 56 28 0 86 200 0 286 1054 0 ...
##  $ cs_000: int  10476 6160 3584 1032 3942 3812 1228 7650 11526 556 ...
##  $ cs_001: int  1226 796 500 6 520 530 20 950 1340 62 ...
##  $ cs_002: int  267998 164860 56362 24 80950 73228 176 59058 202326 11606 ...
##  $ cs_003: int  521832 350066 149726 656 227322 155666 654 298814 1441484 370336 ...
##  $ cs_004: int  428776 272956 100326 692 186242 95538 556 126308 598470 119892 ...
##  $ cs_005: int  4015854 1837600 1744838 4836 2288268 1874972 1402 4436996 3676070 399936 ...
##  $ cs_006: int  895240 301242 488302 388 1137268 443538 1474 1138924 1196806 149156 ...
##  $ cs_007: int  26330 9148 16682 0 22228 15672 608 37752 36184 2826 ...
##  $ cs_008: int  118 22 246 0 204 86 0 352 170 0 ...
##  $ cs_009: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ee_000: int  965866 664504 262032 5670 404740 301078 4252 543294 1875574 512878 ...
##  $ ee_001: int  1706908 824154 453378 1566 904230 481542 1562 907360 1759758 293880 ...
##  $ ee_002: int  1240520 421400 277378 240 622012 388574 168 715518 699290 129862 ...
##  $ ee_003: int  493384 178064 159812 46 229790 288278 48 384948 362510 26872 ...
##  $ ee_004: int  721044 293306 423992 58 405298 900430 60 915978 1190028 34044 ...
##  $ ee_005: int  469792 245416 409564 44 347188 300412 28 1052166 1012704 22472 ...
##  $ ee_006: int  339156 133654 320746 10 286954 1534 0 1108672 160090 34362 ...
##  $ ee_007: int  157956 81140 158022 0 311560 338 0 341532 63216 0 ...
##  $ ee_008: int  73224 97576 95128 0 433954 856 0 129504 41202 0 ...
```

```
##  $ ee_009: int  0 1500 514 0 1218 0 0 7832 4 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Para entender que información contiene estos atributos binarizados se mostrará uno de ellos para el primer camión del dataset.

```
histogram_1 <- as.numeric(histogram[1, c(1:10)])
barplot(histogram_1)
```



Si se suman todos los valores del atributo se obtiene el número total de sucesos.

```
sum(histogram_1)
```

```
## [1] 6167850
```

Al hacer la suma en otro atributo también se obtiene el mismo número total de sucesos.

```
histogram_2 <- as.numeric(histogram[1, c(11:20)])
sum(histogram_2)
```

```
## [1] 6167850
```

Esto nos hace deducir, que la suma de todos los valores del histograma representa la cantidad de tiempo que el camión ha estado encendido a lo largo de su vida.

Si por ejemplo un atributo es la temperatura, y se divide en 10 intervalos ([-20º,-10º], [-10º, 0º], ... , [70º, 80º]), tendremos la cantidad de tiempo (por ejemplo minutos) que el camión ha estado en cada intérvalo de temperatura a lo largo de su vida. Al sumar los valores de todos los intérvalos, se obtiene entonces el tiempo total que el camión ha estado encendido a lo largo de su vida.

Por eso la suma, en todos los atributos da siempre el mismo resultado, porque cada atributo se ha monitorizado durante el tiempo que el camión está encendido.

Esto nos hace pensar que los ceros en los histogramas son valores que aportan información útil, y no se deben eliminar o imputar. Ya que si por ejemplo un camión ha estado mucho tiempo en temperaturas extremas, puede afectar gravemente al sistema APS, pero si nunca ha estado en temperaturas extremas, puede que el sistema APS esté en buen estado.

Pasamos a tratar los ceros de los atributos no binarizados.

```
other <- select(ctrain_km, -colnames(histogram))
sort(colMeans(other=="0"), decreasing = TRUE)
```

```
##      as_000      au_000      dj_000      dl_000      dm_000      ef_000
## 0.999664688 0.998943768 0.996378634 0.996076853 0.995473292 0.995439761
##      dz_000      dk_000      ak_000      eg_000      ea_000      df_000
## 0.994718841 0.994282936 0.991835161 0.991365724 0.986973142 0.980803407
##      dg_000      ae_000      af_000      ar_000      ai_000      at_000
## 0.972269725 0.966099990 0.964809040 0.954481440 0.902357241 0.899440030
##      dh_000      di_000      aj_000      cj_000      dq_000      dr_000
## 0.871843879 0.813063743 0.790094893 0.789793113 0.771820407 0.769204976
##      cl_000      dx_000      dy_000      al_000       am_0      cm_000
## 0.728649029 0.686416524 0.632163096 0.631861315 0.628256715 0.490644804
##      bf_000      eb_000      bc_000      bz_000      do_000      dp_000
## 0.432920900 0.374073702 0.332796835 0.235489387 0.226033598 0.216678403
##      ce_000      ax_000      av_000      cp_000      ac_000      be_000
## 0.212721725 0.181370083 0.178301982 0.171914294 0.159172451 0.118717098
##      bd_000      du_000      dv_000      ds_000      dt_000      ed_000
## 0.069543641 0.056600610 0.056265299 0.045753278 0.043842001 0.029691849
##       ec_00      by_000      de_000      dd_000      cc_000      aa_000
## 0.027042886 0.020537840 0.015776414 0.015659055 0.015625524 0.005985313
##      aq_000      ci_000      ck_000      ao_000      bj_000      ca_000
## 0.003872850 0.002917212 0.002431010 0.002363947 0.002296885 0.002280119
##      ah_000      bg_000      bh_000      an_000      bt_000      bs_000
## 0.002229823 0.002229823 0.002196291 0.002179526 0.002162760 0.002028636
##      bi_000      ap_000      cb_000      bu_000      bb_000      cq_000
## 0.001995104 0.001911277 0.001877745 0.001844214 0.001810683 0.001810683
##      bv_000      dn_000       class      bx_000      cd_000
## 0.001793917 0.001793917 0.000000000 0.000000000 0.000000000
```

Se puede apreciar que algunos atributos no binarizados tienen más de un 90% de ceros (en concreto hay 10 atributos que tienen más de un 99% de ceros). Procedemos a eliminar los atributos con más de un 90% de ceros.

```
columns_to_remove <- which(colMeans(other=="0") > 0.9)
ctrain_km = subset(ctrain_km, select = -c(columns_to_remove) )
dim(ctrain_km)
```

```
## [1] 59646   130
```

Se identifican también las variables sin variabilidad, aquellas de único valor. Se obtiene que la variable cd_000 presenta un único valor. Se procede a su eliminación.

```
# Ver valores únicos
apply(ctrain_km, 2, function(x) length(unique(x)))
```

```
##  class aa_000 ac_000 ag_000 ag_002 ag_004 ag_005 ag_006 ag_007 ag_008 ag_009
##      2  21960   2061    153   2420  23069  40795  40613  32124  18931   5355
## aj_000 al_000   am_0 an_000 ao_000 ap_000 aq_000 ar_000 as_000 at_000 au_000
##    889   8990  10552  48083  47607  44572  34863     68     20   3504     56
## av_000 ax_000 ay_000 ay_001 ay_002 ay_003 ay_004 ay_005 ay_006 ay_007 ay_008
##   3859   2214    466    919    986   1032   1767  19696  34628  38651  37822
## ay_009 az_000 az_001 az_002 az_003 az_004 az_005 az_006 az_007 az_008 az_009
##    447   9184   7378   9064  21530  34077  44601  12306   3876   1271    336
## ba_002 ba_003 ba_008 ba_009 bb_000 bc_000 bd_000 be_000 bf_000 bg_000 bh_000
##  35956  32912  11815   6796  49267   2806   3562   3913   1094  43477  25202
## bi_000 bj_000 bu_000 bv_000 bx_000 ca_000 cb_000 cc_000 cd_000 ce_000 ci_000
##  41995  38287  49226  49223  52807  28280  30115  43825      1  22053  45878
## cj_000 ck_000 cl_000 cm_000 cn_000 cn_001 cn_002 cn_003 cn_004 cn_005 cn_006
##   7595  44956    969   2120   1529   5417  14787  34108  42084  38552  32424
## cn_007 cn_008 cn_009 cp_000 cq_000 cs_000 cs_001 cs_002 cs_003 cs_004 cs_005
##  21995  10117   3055   2313  49224   9295   3341  28276  35191  34556  42556
## cs_006 cs_007 cs_008 cs_009 dd_000 de_000 df_000 dg_000 dh_000 di_000 dj_000
##  40914  17299    773     53   6554   1893    396   1300   1052   5606     72
## dk_000 dl_000 dm_000 dn_000 do_000 dp_000 dq_000 dr_000 ds_000 dt_000 du_000
##    256    181    221  21069  20496  11561   8061   6674  26295  15898  28471
## dv_000 dx_000 dy_000 dz_000 ea_000 eb_000  ec_00 ed_000 ee_000 ee_001 ee_002
##  30112  14695   6304     47    134  27778  31125   3924  41815  38245  34488
## ee_003 ee_004 ee_005 ee_006 ee_007 ee_008 ee_009 ef_000 eg_000
##  31711  35188  36288  31795  30469  24213   9724     28     49
```

Se puede ver como el atributo cd_000 contiene un único valor para el data set, por lo tanto desechamos este atributo.

```
ctrain_km <- subset(ctrain_km, select=-c(cd_000))
dim(ctrain_km)
```

```
## [1] 59646   129
```

Este sería el conjunto de datos de train con imputación de perdidos por km (train_km) limpio y preparado para el análisis. Se seleccionan las mismas variables para el conjunto train con imputación por mediana (train_median), así como para los conjuntos de test.

```
# Conjunto de datos de train y test según método de imputación, preparados para análisis
ctrain_median<-select(ctrain_median, colnames(ctrain_km))
ctest_km <- select(ctest_km, colnames(ctrain_km))
ctest_median <- select(ctest_median, colnames(ctrain_km))
dim(ctest_km)
```

```
## [1] 16000   129
```

En este punto se realiza una copia de ambos conjuntos de datos para el inicio del análisis.

```
train_km<-ctrain_km
train_median<-ctrain_median
test_km<-ctest_km
test_median<-ctest_median
```

Se seleccionan los datos numéricos y se normalizan. Después de la normalización se vuelve a hacer un screening de datos.

```
# Escalado para datos con km
class(train_km)
```

```
## [1] "data.table" "data.frame"
```

```
train_km_X.scaled <- scale(train_km[,c(2:129)])
test_km_X.scaled <- scale(test_km[,c(2:129)])
#eda_skim<-skim(train_km_X.scaled)
#eda_skim

# Escalado para datos con median
train_median_X.scaled <- scale(train_median[,c(2:129)])
test_median_X.scaled <- scale(test_median[,c(2:129)])
```

Con los datos normalizados se hace una reducción de dimensionalidad mediante PCA. Para los datos km el 95% de la varianza queda explicada con 71 componentes principales, mientras que para los datos median el 95% de la varianza queda explicada con 66 componentes principales.

```
# PCA0 con datos km
library(pcaMethods)
```

```
## Loading required package: Biobase
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
##
## Attaching package: 'pcaMethods'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
train_km.pca <- prcomp(train_km_X.scaled[,c(1:(dim(train_km)[2]-1))], center = TRUE, scale = TRUE)
test_km.pca <- prcomp(test_km_X.scaled[,c(1:(dim(test_km)[2]-1))], center = TRUE, scale = TRUE)
summary(train_km.pca)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5    PC6     PC7
## Standard deviation     6.5231 2.45561 1.99079 1.87811 1.78138 1.6970 1.47102
## Proportion of Variance 0.3324 0.04711 0.03096 0.02756 0.02479 0.0225 0.01691
## Cumulative Proportion  0.3324 0.37954 0.41050 0.43806 0.46285 0.4854 0.50225
##                           PC8     PC9   PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.43249 1.37259 1.3434 1.33010 1.30744 1.28813 1.26680
## Proportion of Variance 0.01603 0.01472 0.0141 0.01382 0.01335 0.01296 0.01254
## Cumulative Proportion  0.51829 0.53300 0.5471 0.56092 0.57428 0.58724 0.59978
##                           PC15    PC16    PC17    PC18   PC19    PC20    PC21
## Standard deviation     1.25050 1.18433 1.15412 1.14132 1.1257 1.11041 1.09310
## Proportion of Variance 0.01222 0.01096 0.01041 0.01018 0.0099 0.00963 0.00933
## Cumulative Proportion  0.61200 0.62295 0.63336 0.64354 0.6534 0.66307 0.67240
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     1.0852 1.06903 1.05261 1.04404 1.02956 1.01994 1.01399
## Proportion of Variance 0.0092 0.00893 0.00866 0.00852 0.00828 0.00813 0.00803
## Cumulative Proportion  0.6816 0.69053 0.69919 0.70771 0.71599 0.72411 0.73215
##                           PC29    PC30   PC31    PC32    PC33   PC34    PC35
## Standard deviation     1.00509 1.00018 0.9930 0.98839 0.97719 0.9731 0.96742
## Proportion of Variance 0.00789 0.00782 0.0077 0.00763 0.00746 0.0074 0.00731
## Cumulative Proportion  0.74004 0.74785 0.7556 0.76319 0.77065 0.7781 0.78536
##                           PC36    PC37    PC38    PC39    PC40    PC41   PC42
## Standard deviation     0.96362 0.94967 0.92109 0.92071 0.90265 0.89583 0.8835
## Proportion of Variance 0.00725 0.00705 0.00663 0.00662 0.00637 0.00627 0.0061
## Cumulative Proportion  0.79262 0.79966 0.80629 0.81291 0.81928 0.82555 0.8316
##                           PC43    PC44    PC45    PC46   PC47    PC48    PC49
## Standard deviation     0.8765 0.86945 0.85652 0.84157 0.8313 0.81872 0.81313
## Proportion of Variance 0.0060 0.00591 0.00573 0.00553 0.0054 0.00524 0.00517
## Cumulative Proportion  0.8377 0.84355 0.84928 0.85482 0.8602 0.86545 0.87062
##                           PC50   PC51    PC52    PC53   PC54    PC55    PC56
## Standard deviation     0.80726 0.7997 0.78650 0.76104 0.7508 0.72677 0.71505
## Proportion of Variance 0.00509 0.0050 0.00483 0.00452 0.0044 0.00413 0.00399
## Cumulative Proportion  0.87571 0.8807 0.88554 0.89006 0.8945 0.89859 0.90259
##                           PC57   PC58    PC59    PC60    PC61    PC62    PC63
## Standard deviation     0.71276 0.6972 0.68207 0.67259 0.66788 0.65680 0.64862
## Proportion of Variance 0.00397 0.0038 0.00363 0.00353 0.00348 0.00337 0.00329
## Cumulative Proportion  0.90656 0.9104 0.91399 0.91752 0.92101 0.92438 0.92767
##                           PC64    PC65    PC66    PC67    PC68    PC69    PC70
## Standard deviation     0.64308 0.62574 0.62337 0.61146 0.60221 0.59925 0.59188
## Proportion of Variance 0.00323 0.00306 0.00304 0.00292 0.00283 0.00281 0.00274
## Cumulative Proportion  0.93090 0.93396 0.93699 0.93991 0.94275 0.94555 0.94829
##                           PC71    PC72    PC73    PC74    PC75    PC76    PC77
## Standard deviation     0.58284 0.57504 0.56473 0.54727 0.53979 0.52599 0.51448
## Proportion of Variance 0.00265 0.00258 0.00249 0.00234 0.00228 0.00216 0.00207
## Cumulative Proportion  0.95094 0.95353 0.95602 0.95836 0.96063 0.96280 0.96486
##                           PC78    PC79    PC80    PC81    PC82    PC83   PC84
## Standard deviation     0.51321 0.49455 0.48184 0.48095 0.46957 0.46143 0.4527
## Proportion of Variance 0.00206 0.00191 0.00181 0.00181 0.00172 0.00166 0.0016
## Cumulative Proportion  0.96692 0.96883 0.97065 0.97245 0.97418 0.97584 0.9774
##                           PC85   PC86    PC87    PC88    PC89    PC90    PC91
## Standard deviation     0.44837 0.4384 0.43221 0.42737 0.42094 0.41531 0.40973
## Proportion of Variance 0.00157 0.0015 0.00146 0.00143 0.00138 0.00135 0.00131
## Cumulative Proportion  0.97901 0.9805 0.98197 0.98340 0.98478 0.98613 0.98744
##                           PC92    PC93   PC94    PC95    PC96   PC97    PC98
## Standard deviation     0.39383 0.37630 0.3747 0.35303 0.34870 0.3398 0.32471
## Proportion of Variance 0.00121 0.00111 0.0011 0.00097 0.00095 0.0009 0.00082
## Cumulative Proportion  0.98865 0.98976 0.9909 0.99183 0.99278 0.9937 0.99451
##                           PC99   PC100  PC101   PC102   PC103   PC104   PC105
## Standard deviation     0.28047 0.25607 0.2536 0.24470 0.22467 0.21777 0.20189
## Proportion of Variance 0.00061 0.00051 0.0005 0.00047 0.00039 0.00037 0.00032
## Cumulative Proportion  0.99512 0.99563 0.9961 0.99660 0.99700 0.99737 0.99769
##                          PC106   PC107   PC108   PC109   PC110   PC111   PC112
## Standard deviation     0.1950 0.18875 0.16978 0.16384 0.14048 0.13446 0.13215
## Proportion of Variance 0.0003 0.00028 0.00023 0.00021 0.00015 0.00014 0.00014
## Cumulative Proportion  0.9980 0.99826 0.99849 0.99870 0.99885 0.99899 0.99913
##                           PC113   PC114   PC115  PC116   PC117   PC118   PC119
## Standard deviation     0.12447 0.12176 0.11797 0.1114 0.10393 0.09983 0.09032
## Proportion of Variance 0.00012 0.00012 0.00011 0.0001 0.00008 0.00008 0.00006
## Cumulative Proportion  0.99925 0.99937 0.99947 0.9996 0.99966 0.99973 0.99980
##                           PC120   PC121   PC122   PC123   PC124   PC125   PC126
## Standard deviation     0.08526 0.08121 0.06713 0.05844 0.04284 0.03751 0.02036
```

```
## Proportion of Variance 0.00006 0.00005 0.00004 0.00003 0.00001 0.00001 0.00000
## Cumulative Proportion  0.99985 0.99991 0.99994 0.99997 0.99998 0.99999 1.00000
##                           PC127   PC128
## Standard deviation     0.01732 0.01631
## Proportion of Variance 0.00000 0.00000
## Cumulative Proportion  1.00000 1.00000
```

```
# PCA con datos median
train_median.pca <- prcomp(train_median_X.scaled[,c(1:(dim(train_median)[2]-1))], center = TRUE, scale = TRUE)
test_median.pca <- prcomp(test_median_X.scaled[,c(1:(dim(test_median)[2]-1))], center = TRUE, scale = TRUE)
summary(train_median.pca)
```

```
## Importance of components:
##                            PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     6.3571 2.54570 2.31100 1.99090 1.88342 1.74943 1.61158
## Proportion of Variance 0.3157 0.05063 0.04172 0.03097 0.02771 0.02391 0.02029
## Cumulative Proportion  0.3157 0.36636 0.40808 0.43905 0.46676 0.49067 0.51096
##                            PC8     PC9    PC10   PC11   PC12    PC13    PC14
## Standard deviation     1.45732 1.40770 1.38602 1.3481 1.3385 1.33252 1.28847
## Proportion of Variance 0.01659 0.01548 0.01501 0.0142 0.0140 0.01387 0.01297
## Cumulative Proportion  0.52755 0.54303 0.55804 0.5722 0.5862 0.60011 0.61308
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     1.25855 1.25810 1.20522 1.18182 1.16440 1.13874 1.12305
## Proportion of Variance 0.01237 0.01237 0.01135 0.01091 0.01059 0.01013 0.00985
## Cumulative Proportion  0.62545 0.63782 0.64917 0.66008 0.67067 0.68080 0.69066
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     1.10983 1.07507 1.06363 1.05424 1.04552 1.03946 1.02286
## Proportion of Variance 0.00962 0.00903 0.00884 0.00868 0.00854 0.00844 0.00817
## Cumulative Proportion  0.70028 0.70931 0.71815 0.72683 0.73537 0.74381 0.75198
##                           PC29   PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     1.01440 1.0056 0.99550 0.98836 0.98507 0.97538 0.97125
## Proportion of Variance 0.00804 0.0079 0.00774 0.00763 0.00758 0.00743 0.00737
## Cumulative Proportion  0.76002 0.7679 0.77567 0.78330 0.79088 0.79831 0.80568
##                           PC36    PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.96252 0.95225 0.92648 0.91427 0.89310 0.87858 0.87354
## Proportion of Variance 0.00724 0.00708 0.00671 0.00653 0.00623 0.00603 0.00596
## Cumulative Proportion  0.81292 0.82000 0.82671 0.83324 0.83947 0.84550 0.85146
##                           PC43    PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.85596 0.84510 0.83320 0.82855 0.82152 0.80843 0.80209
## Proportion of Variance 0.00572 0.00558 0.00542 0.00536 0.00527 0.00511 0.00503
## Cumulative Proportion  0.85719 0.86277 0.86819 0.87355 0.87883 0.88393 0.88896
##                           PC50    PC51    PC52   PC53    PC54    PC55    PC56
## Standard deviation     0.78144 0.77167 0.75363 0.7423 0.72516 0.71895 0.69109
## Proportion of Variance 0.00477 0.00465 0.00444 0.0043 0.00411 0.00404 0.00373
## Cumulative Proportion  0.89373 0.89838 0.90282 0.9071 0.91123 0.91527 0.91900
##                           PC57    PC58    PC59    PC60    PC61    PC62    PC63
## Standard deviation     0.68389 0.67626 0.66352 0.65283 0.63776 0.62029 0.60598
## Proportion of Variance 0.00365 0.00357 0.00344 0.00333 0.00318 0.00301 0.00287
## Cumulative Proportion  0.92265 0.92623 0.92967 0.93300 0.93617 0.93918 0.94205
##                           PC64    PC65    PC66    PC67    PC68    PC69   PC70
## Standard deviation     0.60107 0.59348 0.58922 0.57983 0.56042 0.55367 0.5307
## Proportion of Variance 0.00282 0.00275 0.00271 0.00263 0.00245 0.00239 0.0022
## Cumulative Proportion  0.94487 0.94762 0.95034 0.95296 0.95542 0.95781 0.9600
##                           PC71    PC72    PC73   PC74    PC75    PC76    PC77
## Standard deviation     0.52753 0.51357 0.49542 0.4925 0.48490 0.47826 0.46512
## Proportion of Variance 0.00217 0.00206 0.00192 0.0019 0.00184 0.00179 0.00169
## Cumulative Proportion  0.96218 0.96425 0.96616 0.9681 0.96989 0.97168 0.97337
##                           PC78    PC79    PC80    PC81   PC82    PC83    PC84
## Standard deviation     0.46188 0.44486 0.44173 0.43337 0.4229 0.41401 0.40685
## Proportion of Variance 0.00167 0.00155 0.00152 0.00147 0.0014 0.00134 0.00129
## Cumulative Proportion  0.97504 0.97658 0.97811 0.97958 0.9810 0.98231 0.98361
##                           PC85    PC86    PC87    PC88    PC89    PC90    PC91
## Standard deviation     0.39813 0.39550 0.38412 0.37801 0.36933 0.36030 0.34527
## Proportion of Variance 0.00124 0.00122 0.00115 0.00112 0.00107 0.00101 0.00093
## Cumulative Proportion  0.98484 0.98607 0.98722 0.98834 0.98940 0.99042 0.99135
##                           PC92    PC93   PC94    PC95    PC96    PC97    PC98
## Standard deviation     0.3402 0.32482 0.3197 0.31072 0.26887 0.25428 0.24687
## Proportion of Variance 0.0009 0.00082 0.0008 0.00075 0.00056 0.00051 0.00048
## Cumulative Proportion  0.9922 0.99307 0.9939 0.99463 0.99519 0.99570 0.99617
##                           PC99   PC100   PC101   PC102   PC103   PC104   PC105
## Standard deviation     0.24036 0.22416 0.21630 0.20843 0.19809 0.18422 0.16683
## Proportion of Variance 0.00045 0.00039 0.00037 0.00034 0.00031 0.00027 0.00022
## Cumulative Proportion  0.99663 0.99702 0.99738 0.99772 0.99803 0.99829 0.99851
##                          PC106   PC107   PC108   PC109   PC110   PC111   PC112
## Standard deviation     0.16436 0.15075 0.14441 0.13523 0.13070 0.12332 0.12066
## Proportion of Variance 0.00021 0.00018 0.00016 0.00014 0.00013 0.00012 0.00011
## Cumulative Proportion  0.99872 0.99890 0.99906 0.99921 0.99934 0.99946 0.99957
##                          PC113   PC114   PC115   PC116   PC117   PC118   PC119
## Standard deviation     0.1133 0.10085 0.09834 0.08790 0.08416 0.05830 0.04454
## Proportion of Variance 0.0001 0.00008 0.00008 0.00006 0.00006 0.00003 0.00002
## Cumulative Proportion  0.9997 0.99975 0.99983 0.99989 0.99994 0.99997 0.99999
##                          PC120   PC121   PC122   PC123    PC124     PC125 PC126
## Standard deviation     0.03321 0.02088 0.01535 0.01087 0.003453 0.0005281 1e-05
```

```
## Proportion of Variance 0.00001 0.00000 0.00000 0.00000 0.000000 0.0000000 0e+00
## Cumulative Proportion  0.99999 1.00000 1.00000 1.00000 1.000000 1.0000000 1e+00
##                            PC127     PC128
## Standard deviation     3.528e-06 9.769e-07
## Proportion of Variance 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00
```

Nos quedamos únicamente con aquellos componentes que explican un 95% de la varianza.

```
# Componentes PCA datos training km hasta 95%
pct_km_var_explained <- train_km.pca$sdev^2 / sum(train_km.pca$sdev^2)
cumsum(pct_km_var_explained)
```

```
##   [1] 0.3324286 0.3795384 0.4105011 0.4380580 0.4628494 0.4853485 0.5022539
##   [8] 0.5182854 0.5330041 0.5471029 0.5609245 0.5742791 0.5872423 0.5997796
##  [15] 0.6119965 0.6229546 0.6333608 0.6435375 0.6534369 0.6630699 0.6724047
##  [22] 0.6816053 0.6905337 0.6991899 0.7077057 0.7159869 0.7241141 0.7321466
##  [29] 0.7400388 0.7478541 0.7555583 0.7631905 0.7706506 0.7780489 0.7853607
##  [36] 0.7926151 0.7996610 0.8062892 0.8129119 0.8192773 0.8255469 0.8316449
##  [43] 0.8376466 0.8435524 0.8492839 0.8548170 0.8602158 0.8654526 0.8706180
##  [50] 0.8757092 0.8807060 0.8855386 0.8900634 0.8944678 0.8985943 0.9025889
##  [57] 0.9065579 0.9103557 0.9139903 0.9175245 0.9210094 0.9243796 0.9276664
##  [64] 0.9308973 0.9339563 0.9369921 0.9399130 0.9427463 0.9455517 0.9482886
##  [71] 0.9509425 0.9535259 0.9560175 0.9583574 0.9606337 0.9627952 0.9648631
##  [78] 0.9669208 0.9688315 0.9706454 0.9724525 0.9741752 0.9758386 0.9774398
##  [85] 0.9790104 0.9805117 0.9819711 0.9833980 0.9847823 0.9861298 0.9874414
##  [92] 0.9886531 0.9897594 0.9908564 0.9918301 0.9927800 0.9936823 0.9945060
##  [99] 0.9951206 0.9956329 0.9961351 0.9966029 0.9969973 0.9973677 0.9976862
## [106] 0.9979833 0.9982616 0.9984868 0.9986965 0.9988507 0.9989919 0.9991284
## [113] 0.9992494 0.9993652 0.9994740 0.9995708 0.9996552 0.9997331 0.9997968
## [120] 0.9998536 0.9999051 0.9999403 0.9999670 0.9999813 0.9999923 0.9999956
## [127] 0.9999979 1.0000000
```

```
train_km_pca <- data.frame(
    class = train_km$class,
    predict(train_km.pca)[, cumsum(pct_km_var_explained) < 0.95]
)

dim(train_km_pca)
```

```
## [1] 59646    71
```

```
# Componentes PCA datos training median hasta 95%
pct_median_var_explained <- train_median.pca$sdev^2 / sum(train_median.pca$sdev^2)
cumsum(pct_median_var_explained)
```

```
##   [1] 0.3157277 0.3663573 0.4080815 0.4390478 0.4667607 0.4906711 0.5109615
##   [8] 0.5275535 0.5430349 0.5580432 0.5722411 0.5862370 0.6001089 0.6130788
##  [15] 0.6254533 0.6378190 0.6491671 0.6600787 0.6706711 0.6808018 0.6906553
##  [22] 0.7002781 0.7093075 0.7181459 0.7268289 0.7353688 0.7438101 0.7519838
##  [29] 0.7600230 0.7679236 0.7756659 0.7832976 0.7908785 0.7983111 0.8056808
##  [36] 0.8129187 0.8200029 0.8267089 0.8332392 0.8394707 0.8455012 0.8514627
##  [43] 0.8571867 0.8627663 0.8681900 0.8735532 0.8788257 0.8839316 0.8889577
##  [50] 0.8937284 0.8983805 0.9028177 0.9071225 0.9112308 0.9152690 0.9190002
##  [57] 0.9226542 0.9262270 0.9296666 0.9329962 0.9361738 0.9391798 0.9420486
##  [64] 0.9448712 0.9476229 0.9503352 0.9529618 0.9554155 0.9578104 0.9600106
##  [71] 0.9621847 0.9642453 0.9661628 0.9680579 0.9698948 0.9716818 0.9733719
##  [78] 0.9750386 0.9765847 0.9781091 0.9795764 0.9809735 0.9823126 0.9836058
##  [85] 0.9848441 0.9860662 0.9872189 0.9883352 0.9894009 0.9904151 0.9913464
##  [92] 0.9922505 0.9930747 0.9938734 0.9946277 0.9951925 0.9956976 0.9961737
##  [99] 0.9966251 0.9970177 0.9973832 0.9977225 0.9980291 0.9982943 0.9985117
## [106] 0.9987228 0.9989003 0.9990632 0.9992061 0.9993395 0.9994583 0.9995721
## [113] 0.9996724 0.9997518 0.9998274 0.9998877 0.9999431 0.9999696 0.9999851
## [120] 0.9999937 0.9999971 0.9999990 0.9999999 1.0000000 1.0000000 1.0000000
## [127] 1.0000000 1.0000000
```

```
train_median_pca <- data.frame(
    class = train_median$class,
    predict(train_median.pca)[, cumsum(pct_median_var_explained) < 0.95]
)

dim(train_median_pca)
```

```
## [1] 59646    66
```

Seleccionamos los mismos componentes principales para los conjunto de test.

```
# Componentes test km PCA
test_km_pca <- data.frame(
    class = test_km$class,
    predict(test_km.pca)
)
test_km_pca<-select(test_km_pca, colnames(train_km_pca))

# Componentes test median PCA
test_median_pca <- data.frame(
    class = test_median$class,
    predict(test_median.pca)
)
test_median_pca<-select(test_median_pca, colnames(train_median_pca))
```

Limpieza de variables para liberar memoria.

```
ctrain<-NULL
ctrain_km<-NULL
ctrain_median<-NULL
ctrain_num<-NULL
train_km_X.scaled<-NULL
train_km.pca<-NULL
train_median_X.scaled<-NULL
train_median.pca<-NULL
train.km<-NULL
histogram<-NULL
eda_skim<-NULL
other<-NULL
ctest<-NULL
ctest_km<-NULL
ctest_median<-NULL
test_km_X.scaled<-NULL
test_median_X.scaled<-NULL
test_km.pca<-NULL
test.km<-NULL
null_rows<-NULL
```

Aquí terminaría la fase de limpieza y preprocesamiento de datos. Debe comentarse que esta fase y preprocesamiento ha resultado ser bastante costosa, debido al alto número de faltantes. Por otro lado, la implementación de ciertas técnicas de imputación ha resultado impracticable con nuestros recursos de cálculo

El resultado son de esta etapa son conjuntos de datos de training y testing, para dos técnicas diferentes de imputación, vecinos km y mediana. Asimismo, para estos conjuntos se ha realizado un PCA para reducir el número de variables explicativas manteniendo el 95% de la información. El objetivo de esta variedad de conjunto de datos es validar en la fase de análisis cuál de ellas nos da mejores resultados, sobre las técnicas de análisis a aplicar.

5 Análisis de los datos

5.1 Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

Los conjuntos de datos que tenemos son los siguientes:

- Datos de training y testeo con imputación basado en k-Means: Tras limpieza y preprocesamiento, 59646 filas y 129 columnas.
    - PCA con imputación basado en k-Means: Tras aplicación PCA, 59646 filas y 71 columnas.

- Datos de training y testeo con imputación basado en mediana: Tras limpieza y preprocesamiento, 59646 filas y 129 columnas.
    - PCA con imputación basado en mediana: Tras aplicación PCA, 59646 filas y 66 columnas.

Es cierto que la aplicación del PCA dificulta la interpretación de las variables (al ser combinación lineal de las originales), más aun en este caso de Scania donde las variables están anonimizadas. Sin embargo, sí que se ha obtenido una reducción considerable (50% de variables para explicar el 95% de la varianza) del número de variables, aunque el número de variables sigue siendo elevado debido al gran tamaño del conjunto de datos.

El objetivo de este conjunto de datos y, por tanto, del análisis a realizar, es la clasificación de fallo con origen en el APS. Para ello se implementarán las siguientes técnicas:

- Árbol de decisión con el algoritmo c5.0: Se realizará el modelo de árbol de decisión sobre el conjunto de datos de PCA.
- Regresión logística: Se aplicará la regresión logística sobre el conjunto completo de datos tras la limpieza. En concreto, se realizará este ejercicio con los datos no balanceados (como lo son en la realidad), realizando un balanceado aplicando la técnica SMOTE [6] (https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/SMOTE) y utilizando pesos para solventar el no equilibrado de casos negativos y positivos.
- Naive Bayes: Se aplica Naive Bayes balanceado (paquete Rose). El paquete NaiveBayes [7] (https://rdrr.io/cran/naivebayes/) proporciona un algoritmo eficiente para la implantación de Naive Bayes.

La variable que se utilizará para comparar la bondad de los diferentes modelos será el coste total. Al final del apartado de análisis se resumen en una tabla los resultados obtenidos.

5.2 Comprobación de la normalidad y homogeneidad de la varianza

En este apartado se analizarán la normalidad y homogeneidad de la varianza. El gran tamaño de la muestra vuelve a ser un reto para aplicar las técnicas de análisis y visualización. Por ejemplo, no se puede utilizar la función shapiro.test porque el tamaño de la muestra debe ser de un máximo de 5000. Se utiliza la función normality() del paquete dlookr, que realiza el test de Shapiro-Wilks a todas las variables numéricas. Si el tamaño de la muestra es superior a 5000, como es el caso, la función realiza una muestra de tamaño 5000 para calcular el estadístico.

Las variables tienen un valor de significancia menor a 0.05, por lo que se rechaza la normalidad de la distribución. Se utiliza la función plot_normality (para las 5 primeras variables) para visualizar el resultado de normalidad. En el QQPlot se observa la falta de normalidad, con una pendiente casi horizontal en la parte derecha de los residuos. La función plot_normality además ofrece gráficos de transformadas muy interesantes, en concreto se aprecia que la transformada logarítmica mejora apreciablemente la distribución hacia la normalidad.

Se quiere comprobar si las variables de datos con clase negativa y positiva presentan la misma variabilidad. En el **test de Levene también se rechaza la hipótesis de igualdad de varianzas**.

```
if (!require("dlookr")) install.packages("dlookr")
```

```
## Loading required package: dlookr
```

```
## Loading required package: mice
```

```
##
## Attaching package: 'mice'
```

```
## The following objects are masked from 'package:BiocGenerics':
##
##     cbind, rbind
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

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

```
##
## Attaching package: 'dlookr'
```

```
## The following object is masked from 'package:base':
##
##     transform
```

```
library(dlookr)
if (!require("car")) install.packages("car")
```

```
## Loading required package: car
```

```
## Loading required package: carData
```

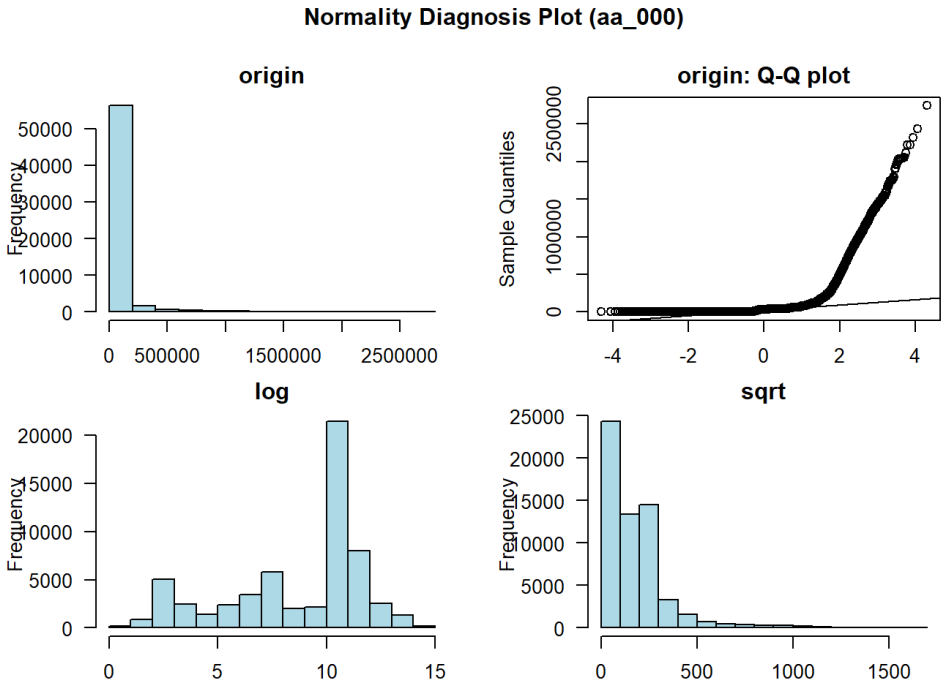
```
##
## Attaching package: 'car'
```

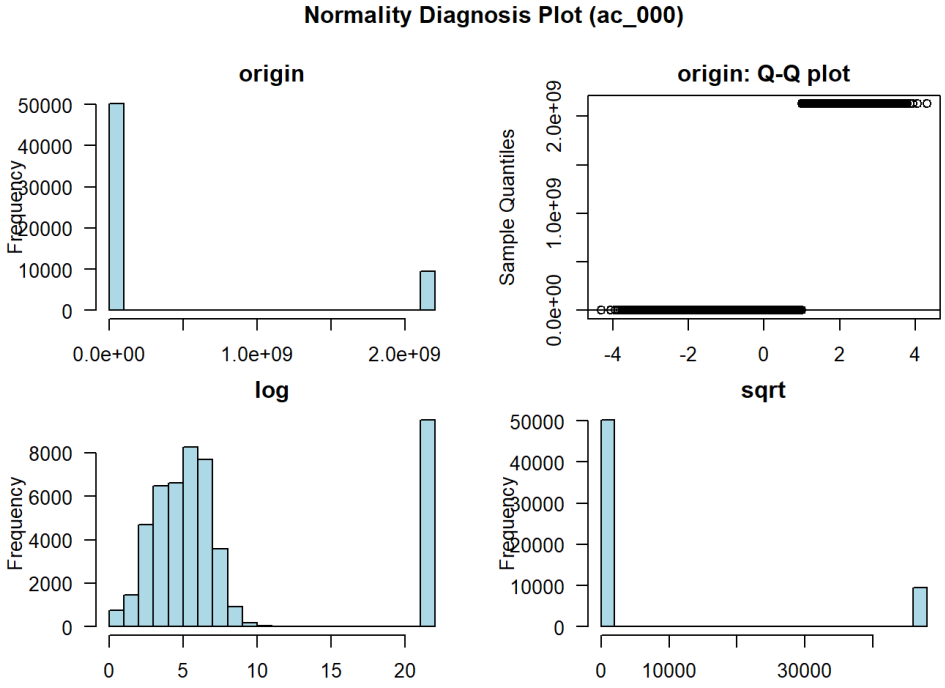
```
## The following object is masked from 'package:dplyr':
##
##     recode
```

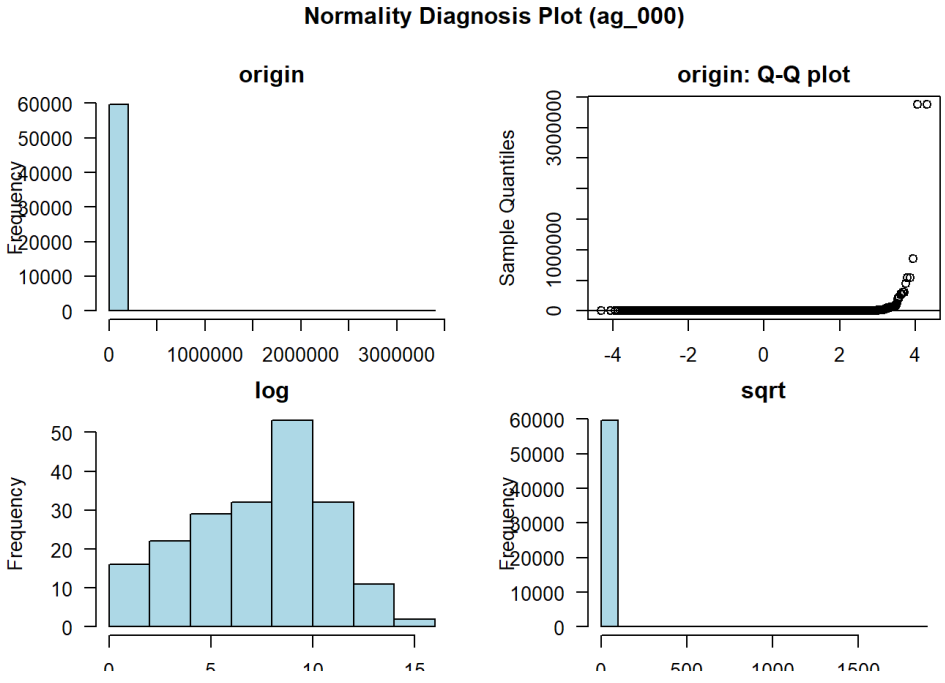
```
library(car)
library(ggplot2)
if (!require("cowplot")) install.packages("cowplot")
```

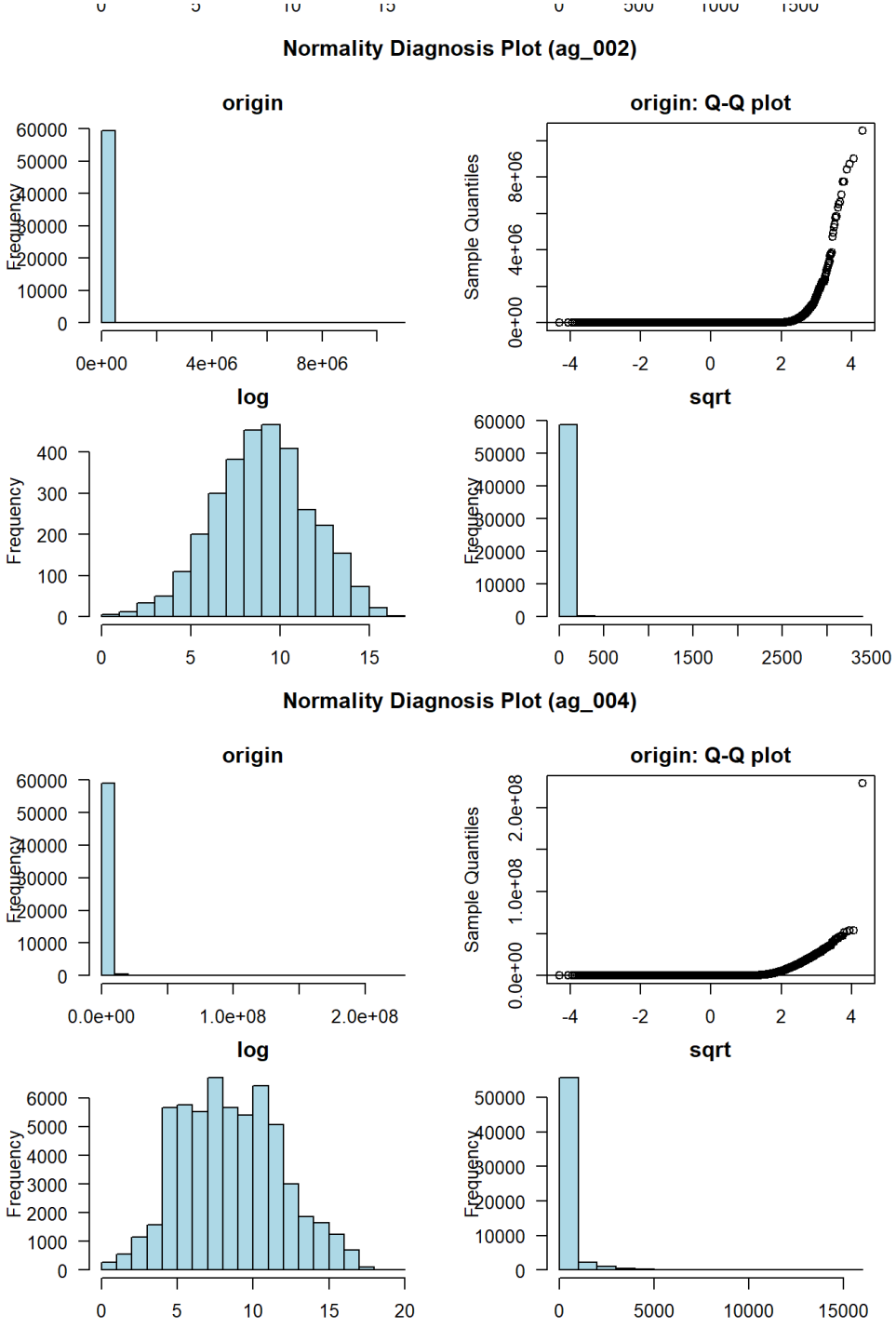
```
## Loading required package: cowplot
```

```
library(cowplot)

train<-train_km
train$class<-as.factor(train$class)

# Visualización de variables numéricas
plot_normality(train[,2:6])
```


Normality Diagnosis Plot (aa_000)
origin
Frequency
origin: Q-Q plot
Sample Quantiles
log
sqrt


Normality Diagnosis Plot (ac_000)
origin
Frequency
origin: Q-Q plot
Sample Quantiles
log
sqrt


Normality Diagnosis Plot (ag_000)
origin
Frequency
origin: Q-Q plot
Sample Quantiles
log
sqrt




```
# Shapiro test para normalidad
trainx<-train[,-1]
sample_rows <- sample( 1:nrow(trainx), 4999 )
train_sample<-trainx[sample_rows,1:50]
train_sample$aa_000<-as.numeric(train_sample$aa_000)
train_sample$ac_000<-as.numeric(train_sample$ac_000)
train_sample$aj_000<-as.numeric(train_sample$aj_000)
train_sample$al_000<-as.numeric(train_sample$al_000)
train_sample$am_0<-as.numeric(train_sample$am_0)
shapiro.test(train_sample$aa_000)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_sample$aa_000
## W = 0.40304, p-value < 2.2e-16
```

```
shapiro.test(train_sample$ac_000)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_sample$ac_000
## W = 0.43635, p-value < 2.2e-16
```

```
shapiro.test(train_sample$aj_000)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_sample$aj_000
## W = 0.010687, p-value < 2.2e-16
```

```
shapiro.test(train_sample$al_000)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_sample$al_000
## W = 0.065367, p-value < 2.2e-16
```

```
shapiro.test(train_sample$am_0)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_sample$am_0
## W = 0.061245, p-value < 2.2e-16
```

```
# Función normality de dlookr que da como resultado para todas las variables de un data frame el test de shapiro
normality_test<-normality(trainx,sample=5000)
head(normality_test,10)
```

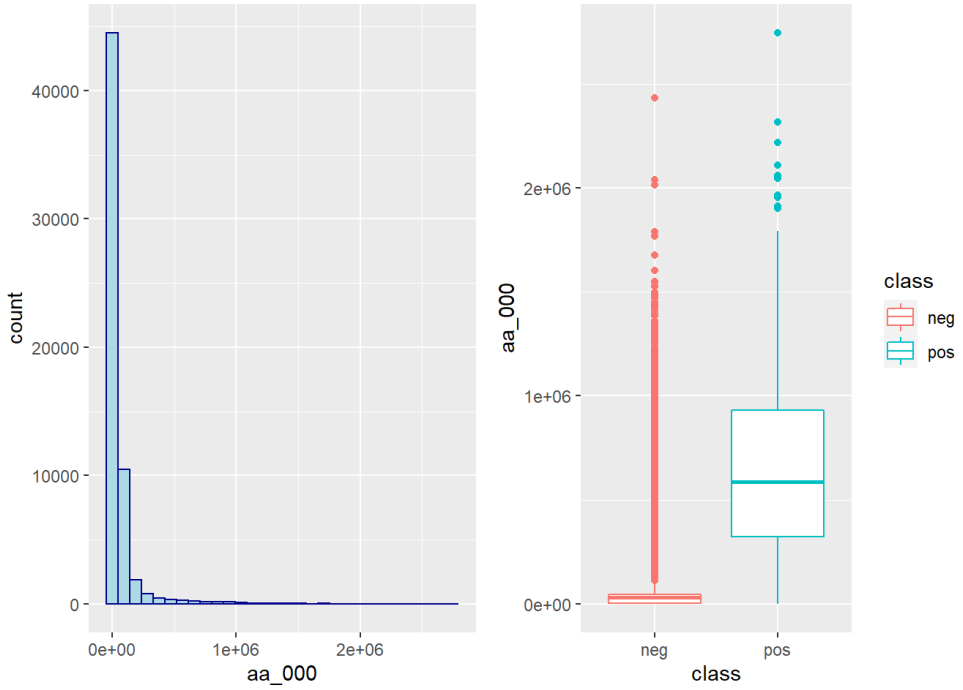
```
## # A tibble: 10 x 4
##    vars   statistic  p_value sample
##    <chr>      <dbl>    <dbl>  <dbl>
##  1 aa_000    0.397  2.88e-84   5000
##  2 ac_000    0.441  1.40e-82   5000
##  3 ag_000    0.0181 2.02e-95   5000
##  4 ag_002    0.0294 3.81e-95   5000
##  5 ag_004    0.208  2.05e-90   5000
##  6 ag_005    0.357  1.07e-85   5000
##  7 ag_006    0.375  4.62e-85   5000
##  8 ag_007    0.326  9.70e-87   5000
##  9 ag_008    0.109  3.89e-93   5000
## 10 ag_009    0.0263 3.20e-95   5000
```

```
levene.tests<-lapply(train[,2:11],function(x) leveneTest(x ~ train$class))
```

Se realizan gráficos para 5 primeras variables, mostrando el histograma de la variable y el boxplot, tanto si la variable corresponde a un caso negativo o positivo.

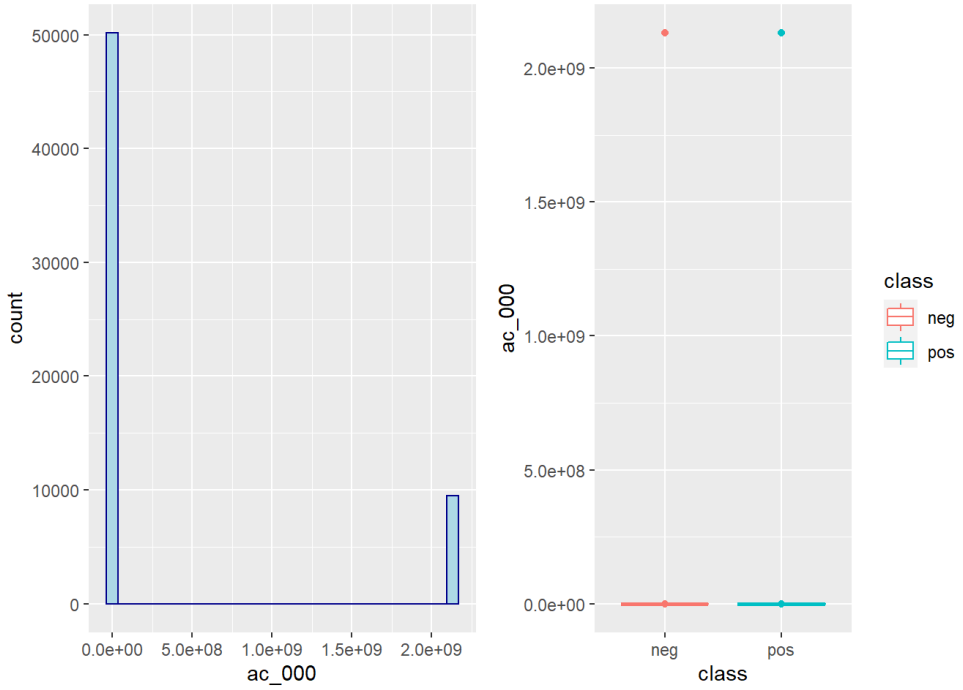
```
# Distribución de aa_000
p1<-ggplot(train, aes(aa_000)) + geom_histogram(color="darkblue", fill="lightblue")
p2<-ggplot(train, aes(x = class, y = aa_000,color=class)) + geom_boxplot(aes(group = class)) + xlab("class") + ylab
("aa_000") + theme_get()
plot_grid(p1, p2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



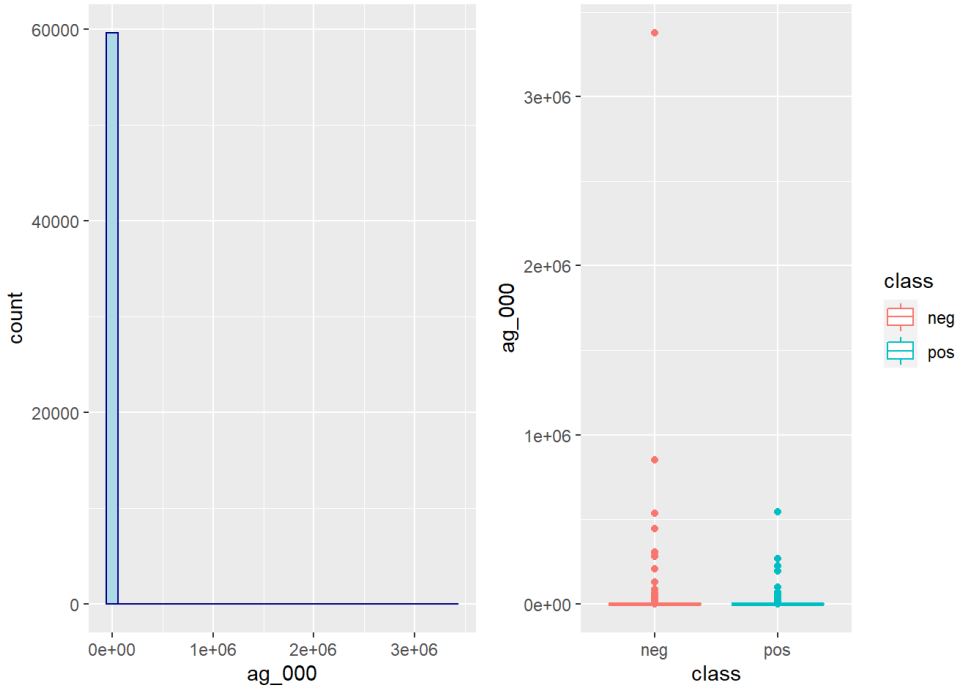

```
# Distribución de ac_000
p1<-ggplot(train, aes(ac_000)) + geom_histogram(color="darkblue", fill="lightblue")
p2<-ggplot(train, aes(x = class, y = ac_000,color=class)) + geom_boxplot(aes(group = class)) + xlab("class") + ylab
("ac_000") + theme_get()
plot_grid(p1, p2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



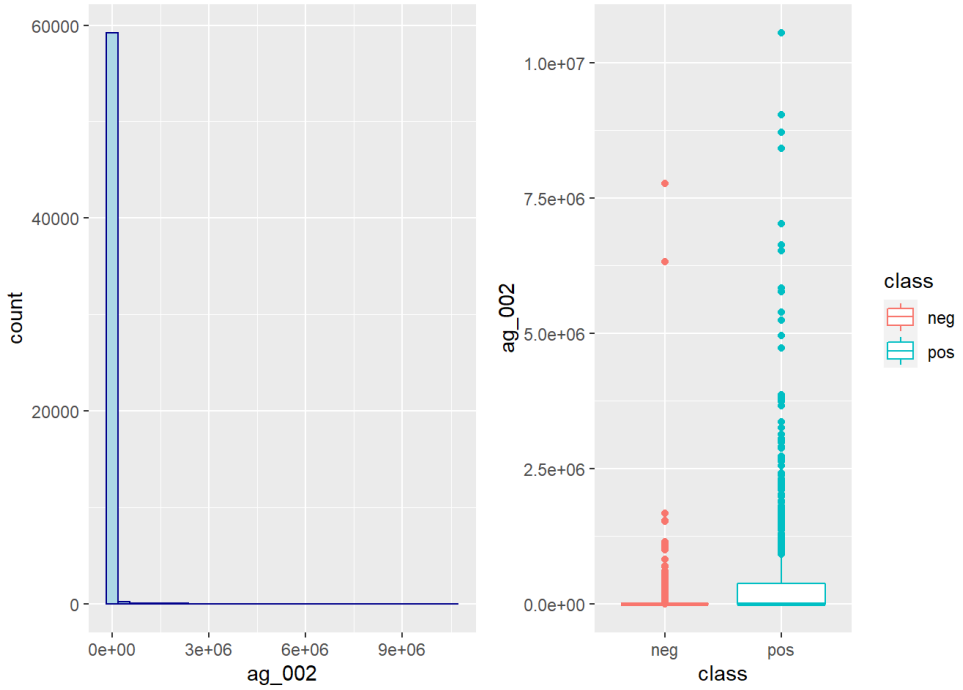

```
# Distribución de ag_000
p1<-ggplot(train, aes(ag_000)) + geom_histogram(color="darkblue", fill="lightblue")
p2<-ggplot(train, aes(x = class, y = ag_000,color=class)) + geom_boxplot(aes(group = class)) + xlab("class") + ylab
("ag_000") + theme_get()
plot_grid(p1, p2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



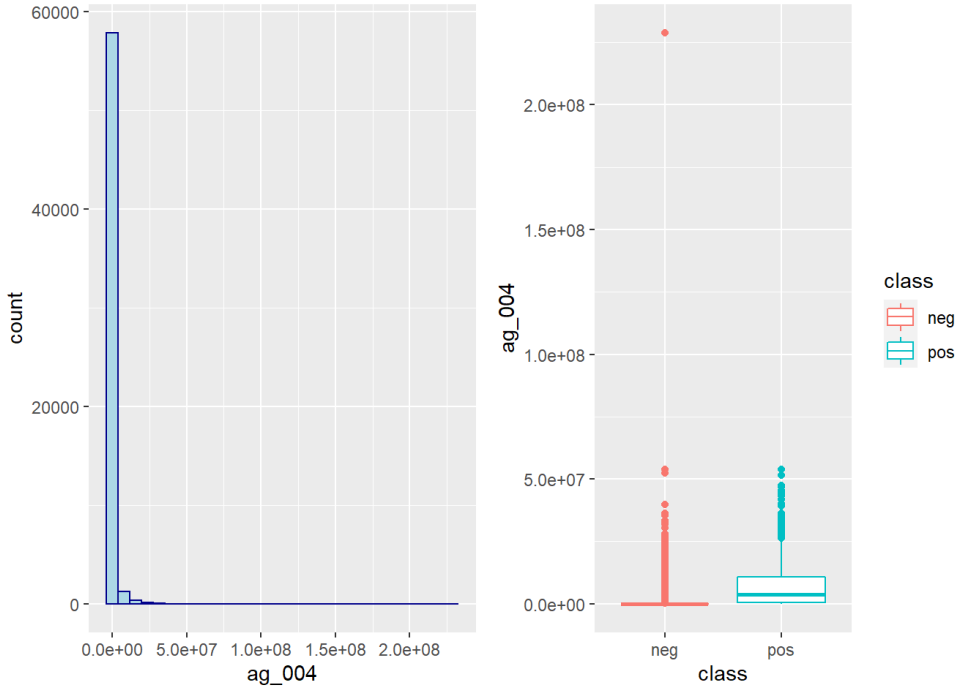

```
# Distribución de ag_002
p1<-ggplot(train, aes(ag_002)) + geom_histogram(color="darkblue", fill="lightblue")
p2<-ggplot(train, aes(x = class, y = ag_002,color=class)) + geom_boxplot(aes(group = class)) + xlab("class") + ylab
("ag_002") + theme_get()
plot_grid(p1, p2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Distribución de ag_004
p1<-ggplot(train, aes(ag_004)) + geom_histogram(color="darkblue", fill="lightblue")
p2<-ggplot(train, aes(x = class, y = ag_004,color=class)) + geom_boxplot(aes(group = class)) + xlab("class") + ylab
("ag_004") + theme_get()
plot_grid(p1, p2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



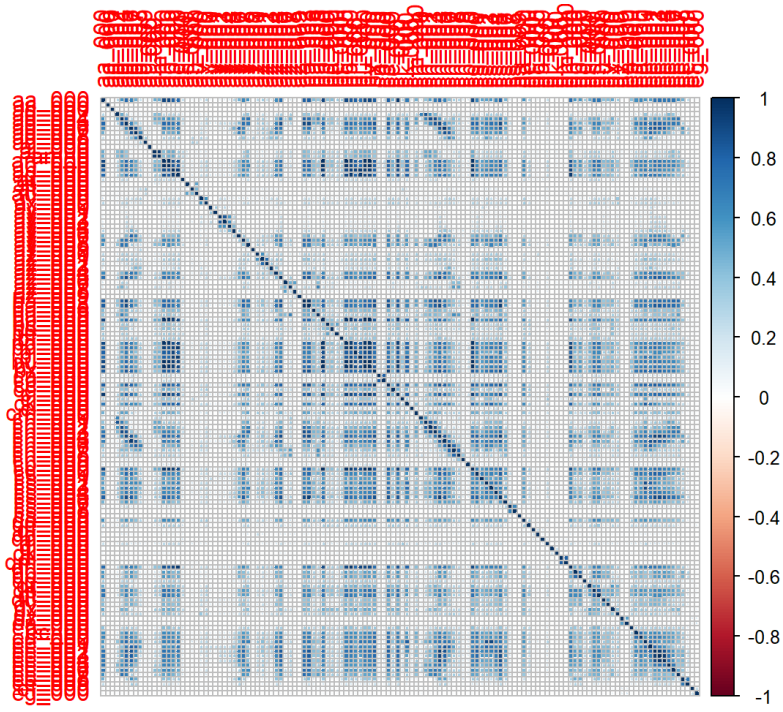

```
p1<-NULL
p2<-NULL
```

Se realizan también un gráfico de correlaciones de todas las variables numéricas. Para ello se utiliza la función corrplot del paquete dlookr, que permite analizar todas las variables numéricas a la vez. Es una función muy útil para análisis exploratorio EDA. Debido al gran tamaño, se vuelve a repetir con las 15 primeras variables, donde claramente se aprecian en color azul las correlaciones positivas. Posteriormente se visualizan en pares de dos (gráfico de dispersión) las relaciones que han resultado ser significativas, aa_000 con an_000, ao_000,ap_000 y aq_000.

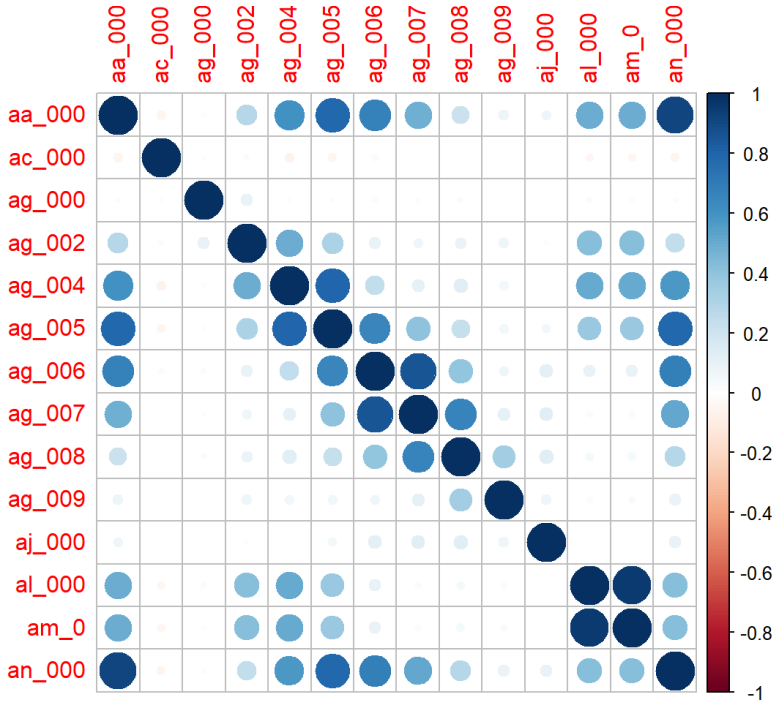
```
# Explotario correlaciones en variables numéricas
library(corrplot)
```

```
## corrplot 0.84 loaded
```

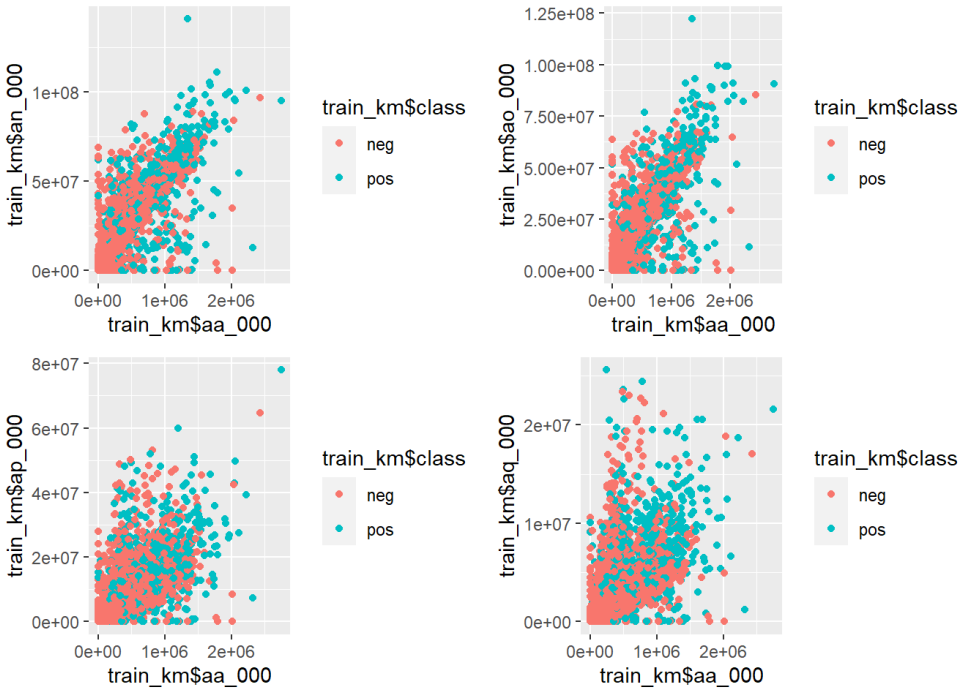
```
## corrplot 0.84 loaded
CP1 <- cor(train_km[,c(2:129)])
corrplot(CP1, method = "circle")
```




```
CP2 <- cor(train_km[,c(2:15)])
corrplot(CP2, method = "circle")
```




```
# Gráficos de dispersión para analizar correlaciones (variables dos a dos)
p1<-ggplot(train_km, aes(x=train_km$aa_000, y=train_km$an_000)) + geom_point(aes(color=train_km$class))
p2<-ggplot(train_km, aes(x=train_km$aa_000, y=train_km$ao_000)) + geom_point(aes(color=train_km$class))
p3<-ggplot(train_km, aes(x=train_km$aa_000, y=train_km$ap_000)) + geom_point(aes(color=train_km$class))
p4<-ggplot(train_km, aes(x=train_km$aa_000, y=train_km$aq_000)) + geom_point(aes(color=train_km$class))
plot_grid(p1,p2,p3,p4,nrow=2)
```



Se han intentado diferentes transformaciones de Box-Cox. Se ha optado por la función Box-Cox del paquete "DescTools", tras analizar los gráficos plot_normality del conjunto de datos train_km. Se ha realizado la transformación logística, con lamda igual a 0. Al no tener que realizar ninguna optimización (si no con tantas variables sería un problema), su ejecución es instantánea. En este caso aparece el problema de los valores infinito (correspondiente al log(0)), que puede traer problemas en la fase de análisis. Se opta por sustituir los valores infinito por 0.5, ya que los valores de las variables superan en su variación la unidad y el infinito en este caso es debido a simple transformación matemática. Se realiza la misma transformación sobre el conjunto de testeo y se guardan los conjuntos de datos como train_bc_km y test_bc_km.

En general, las distribuciones se asemejan más a la normal, excepto en variables con gran cantidad de ceros, como pueden ser la ag_000 o ag_002. En estos casos, incluso con la transformación los resultados no son buenos. En la fase de análisis se comprobará si esta transformación mejora la bondad de los modelos de clasificación y regresión.

```
if (!require("DescTools")) install.packages("DescTools")
```

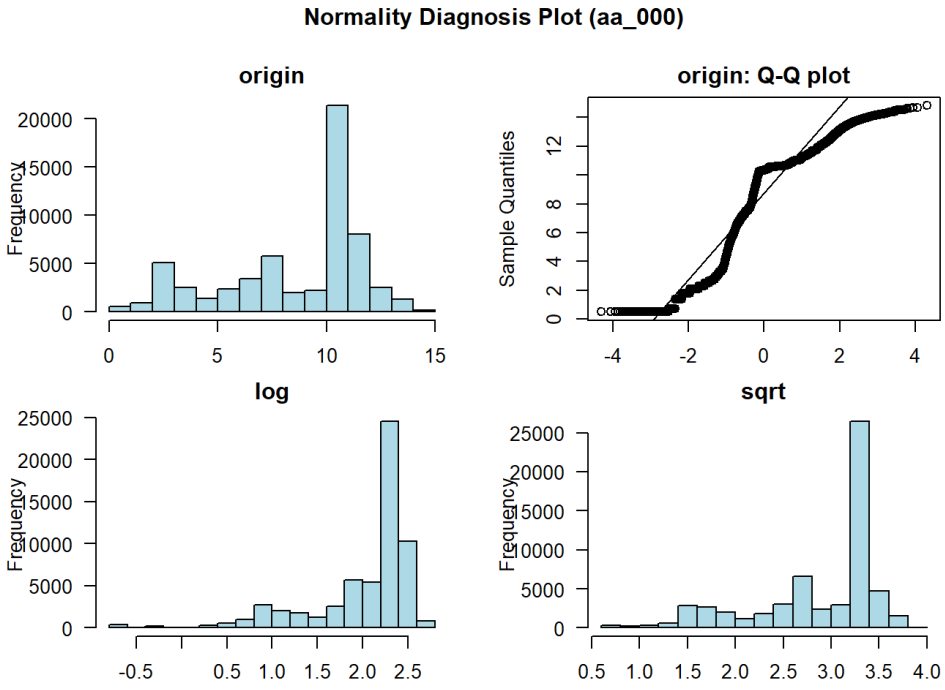
```
## Loading required package: DescTools
```

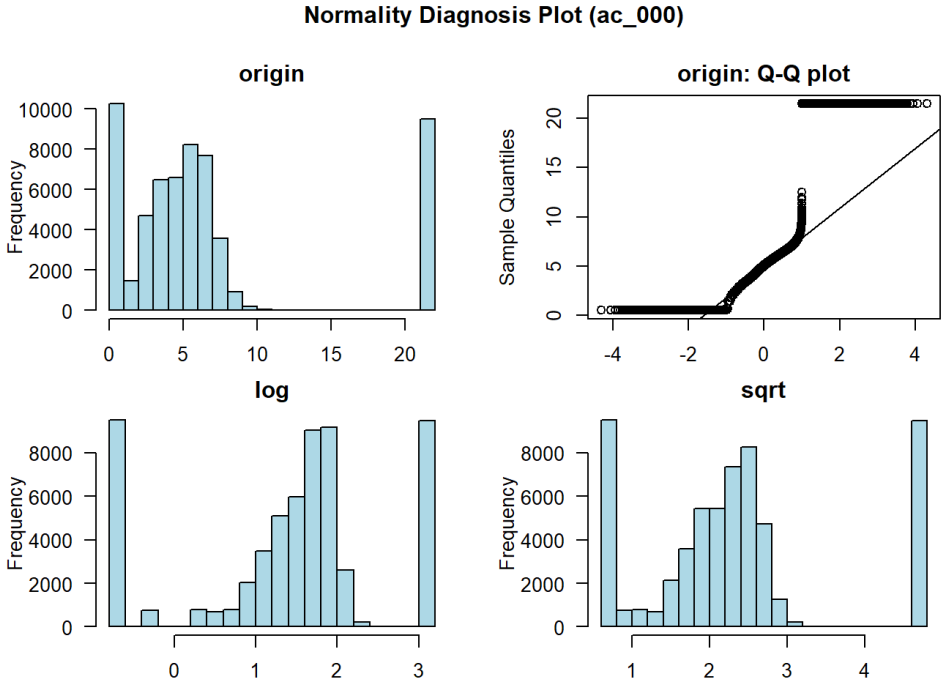
```
##
## Attaching package: 'DescTools'
```

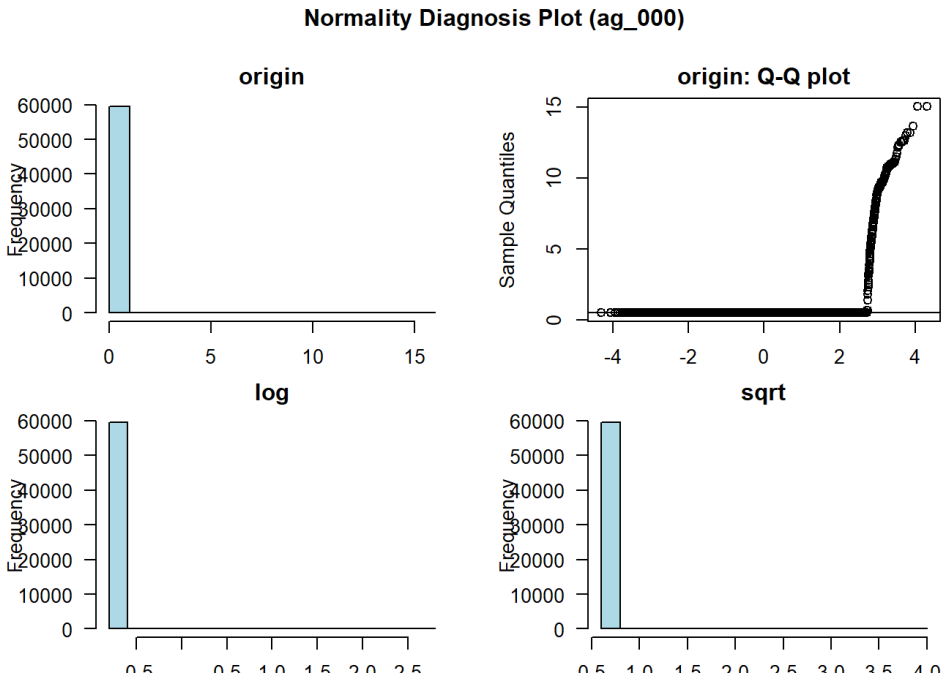
```
## The following object is masked from 'package:car':
##
##     Recode
```

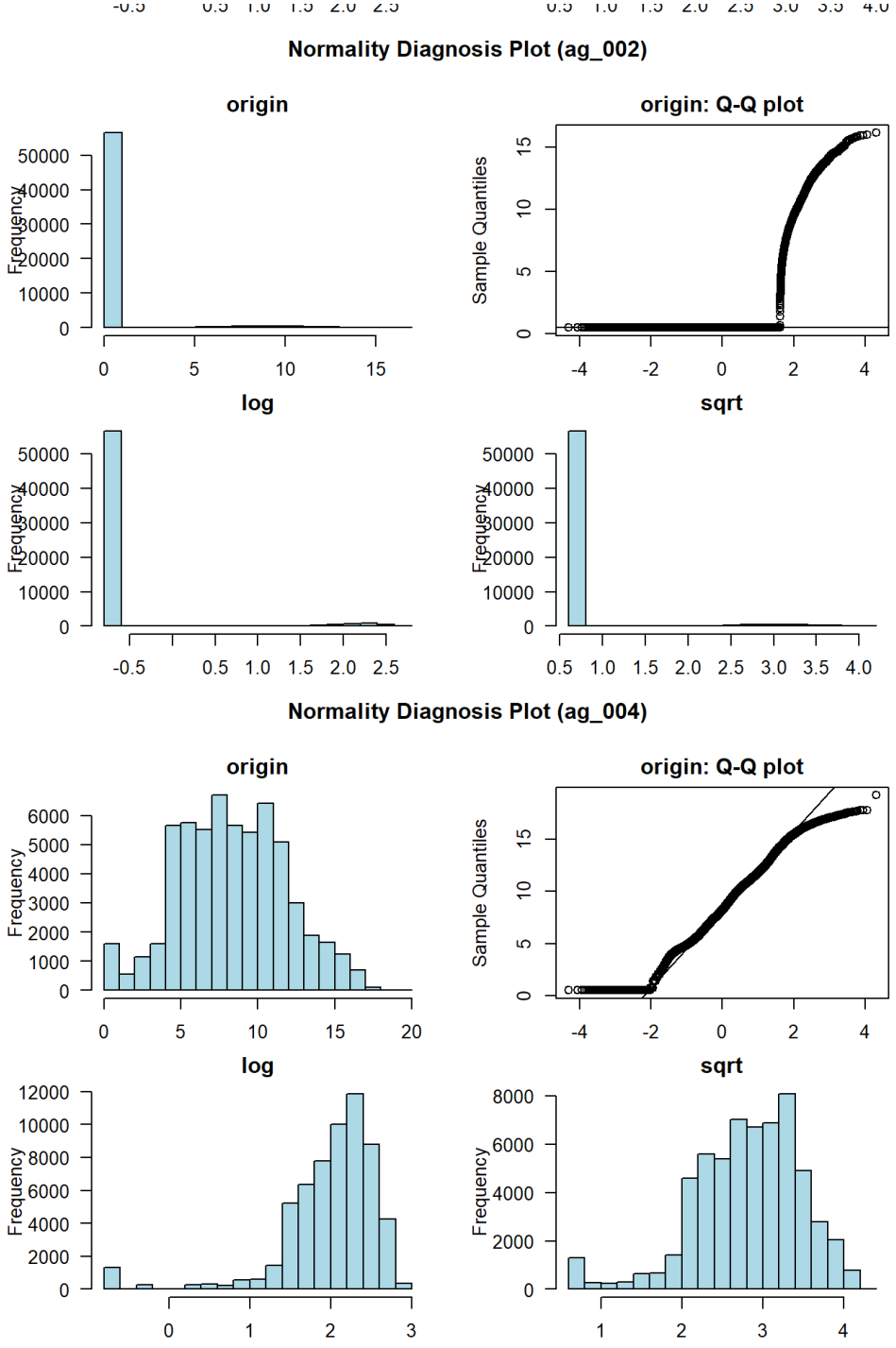
```
## The following object is masked from 'package:data.table':
##
##     %like%
```

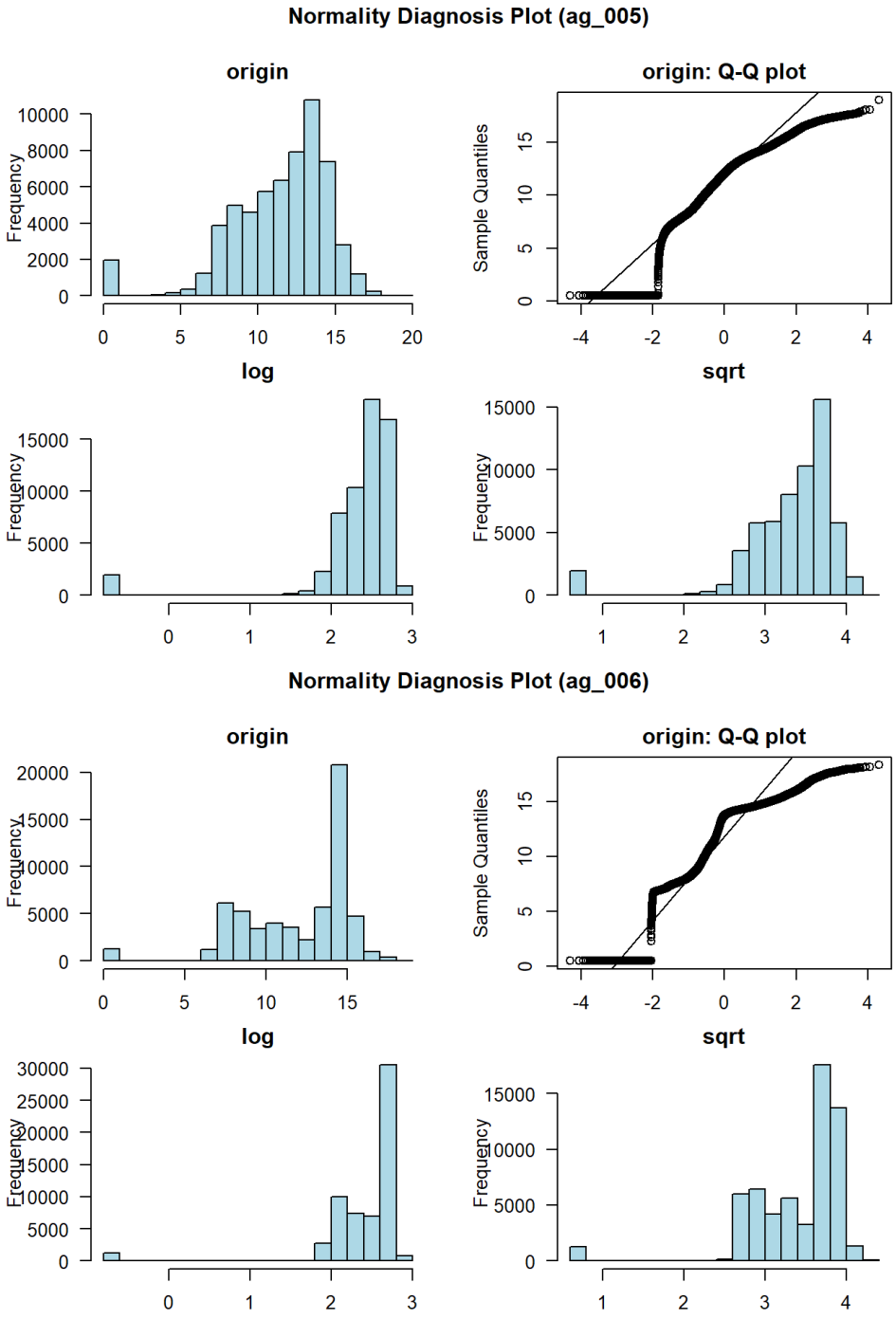
```
library(DescTools)
library(dplyr)
# Transformada de Box-Cox en training (k-means)
x<-NULL
x<-train_km[,c(-1)]
train_bc_km<-NULL
train_bc_km$class<-train_km$class
x<-BoxCox(x, lambda =  0)
x[x==-Inf]<-0.5
plot_normality(x[,1:10])
```

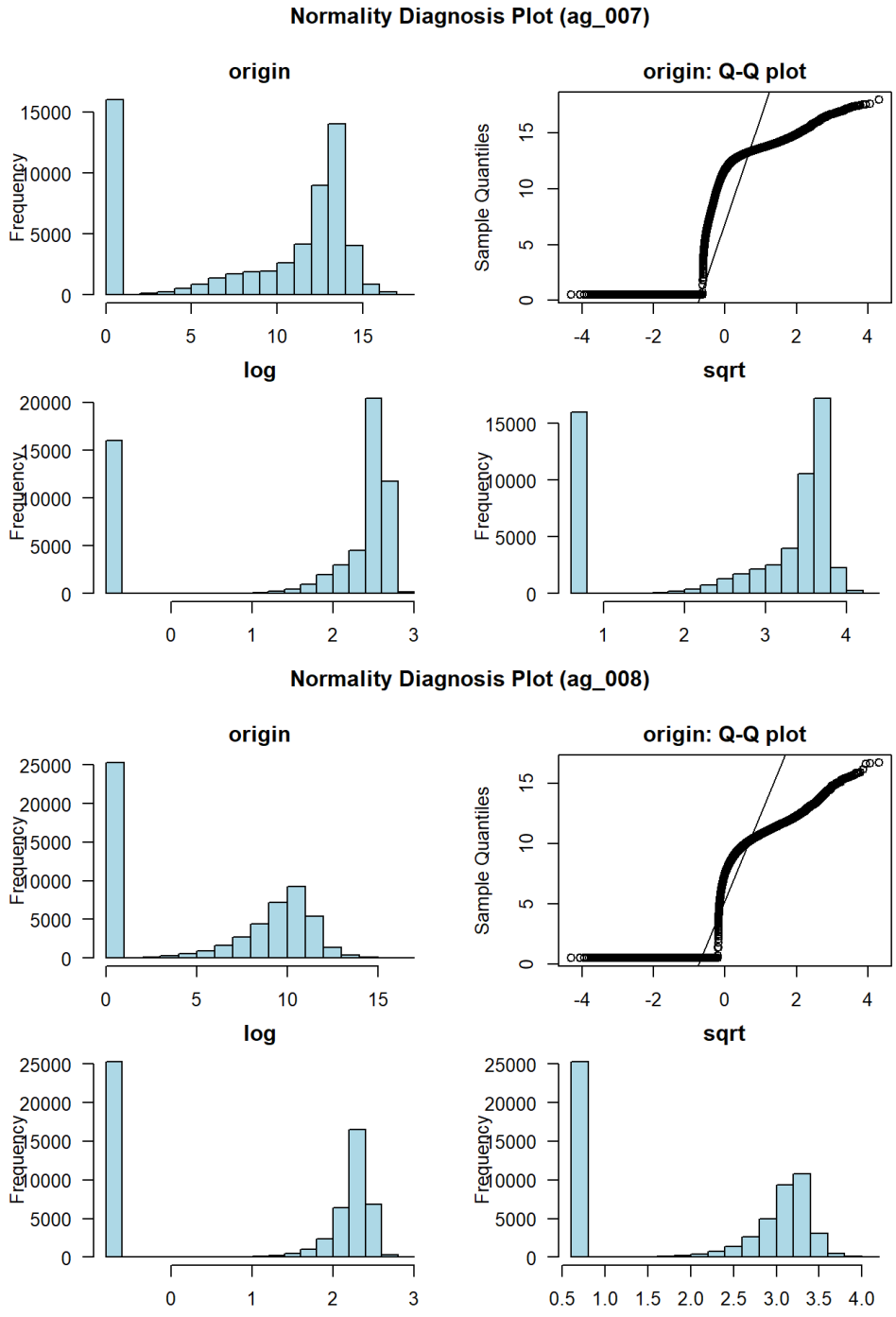

Normality Diagnosis Plot (aa_000)
origin
origin: Q-Q plot
log
sqrt
Frequency
Sample Quantiles

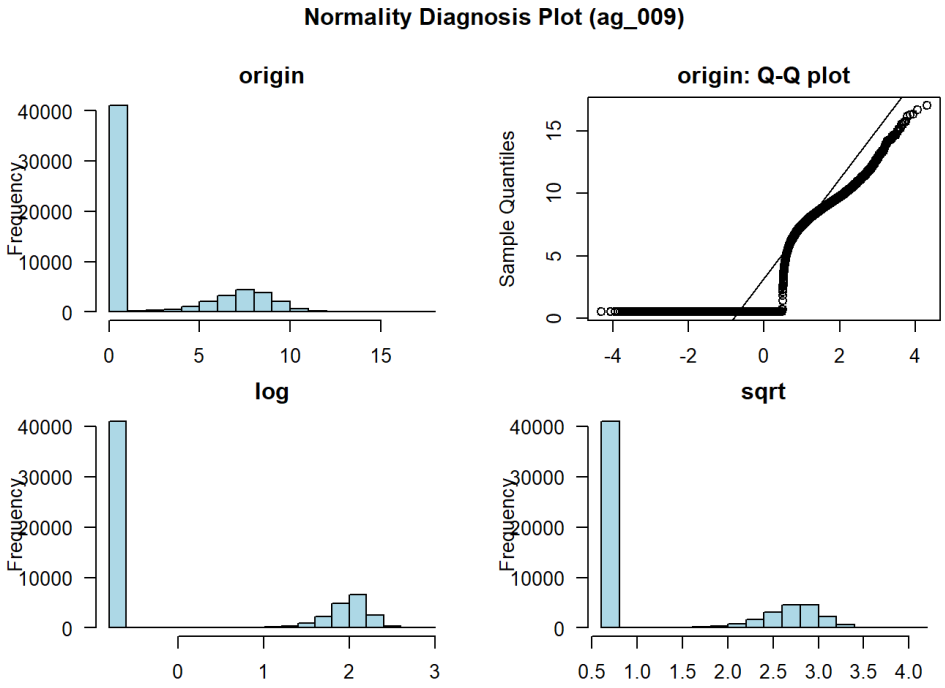

Normality Diagnosis Plot (ac_000)
origin
origin: Q-Q plot
log
sqrt
Frequency
Sample Quantiles


Normality Diagnosis Plot (ag_000)
origin
origin: Q-Q plot
log
sqrt
Frequency
Sample Quantiles


Normality Diagnosis Plot (ag_002)
origin
origin: Q-Q plot
log
sqrt
Frequency
Sample Quantiles
Normality Diagnosis Plot (ag_004)
origin
origin: Q-Q plot
log
sqrt
Frequency
Sample Quantiles


Normality Diagnosis Plot (ag_005)
origin
Frequency
origin: Q-Q plot
Sample Quantiles
log
Frequency
sqrt
Frequency
Normality Diagnosis Plot (ag_006)
origin
Frequency
origin: Q-Q plot
Sample Quantiles
log
Frequency
sqrt
Frequency


Normality Diagnosis Plot (ag_007)
origin
Frequency
origin: Q-Q plot
Sample Quantiles
log
sqrt
Normality Diagnosis Plot (ag_008)
origin
Frequency
origin: Q-Q plot
Sample Quantiles
log
sqrt




```
train_bc_km<-bind_cols(train_bc_km, x)

# Transformada de Box-Cox en testing (k-means)
x<-NULL
x<-test_km[,c(-1)]
test_bc_km<-NULL
test_bc_km$class<-test_km$class
x<-BoxCox(x, lambda =  0)
x[x==-Inf]<-0.5
#plot_normality(x[,1:10])
test_bc_km<-bind_cols(test_bc_km,x)

# Transformada de Box-Cox en training (mediana)
x<-NULL
x<-train_median[,c(-1)]
train_bc_median<-NULL
train_bc_median$class<-train_median$class
x<-BoxCox(x, lambda =  0)
x[x==-Inf]<-0.5
#plot_normality(x[,1:10])
train_bc_median<-bind_cols(train_bc_median, x)

# Transformada de Box-Cox en testing (mediana)
x<-NULL
x<-test_median[,c(-1)]
test_bc_median<-NULL
test_bc_median$class<-test_median$class
x<-BoxCox(x, lambda =  0)
x[x==-Inf]<-0.5
#plot_normality(x[,1:10])
test_bc_median<-bind_cols(test_bc_median,x)
```

5.3 Árbol de decisión con el algoritmo c5.0

5.3.1 c5.0 con datos imputados con k-means

```
# Carga de conjunto de datos
train<-train_km_pca
test<-test_km_pca

# Identificación x e y
trainx <- train[,2:71]
trainy <- as.factor(train[,1])
testx <- test[,2:71]
testy <- as.factor(test[,1])

# Modelo
model <- C50::C5.0(trainx, trainy,rules=TRUE )
summary(model)
```

```
##
## Call:
## C5.0.default(x = trainx, y = trainy, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Jan 04 19:15:33 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 59646 cases (71 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (55759/68, lift 1.0)
##  PC1 > -5.574107
##  PC6 <= 1.813945
##  PC7 <= 2.193364
##  ->  class neg  [0.999]
##
## Rule 2: (56542/127, lift 1.0)
##  PC2 > -2.893974
##  PC4 > -2.765665
##  PC6 <= 1.813945
##  PC7 <= 2.193364
##  PC20 <= 1.399353
##  ->  class neg  [0.998]
##
## Rule 3: (55326/137, lift 1.0)
##  PC4 > -2.765665
##  PC7 <= 2.193364
##  PC34 <= 1.041887
##  PC51 <= 0.3047037
##  PC57 > -2.155799
##  ->  class neg  [0.998]
##
## Rule 4: (55807/190, lift 1.0)
##  PC4 > -2.765665
##  PC9 <= 0.9235685
##  PC15 <= 2.679337
##  PC55 <= 0.4701133
##  ->  class neg  [0.997]
##
## Rule 5: (57486/213, lift 1.0)
##  PC4 > -2.765665
##  PC5 > -1.079978
##  PC7 <= 2.193364
##  ->  class neg  [0.996]
##
## Rule 6: (58798/555, lift 1.0)
##  PC4 > -14.24129
##  PC5 > -8.086918
##  PC6 <= 3.074957
##  PC13 <= 1.989667
##  PC66 > -3.058475
##  ->  class neg  [0.991]
##
## Rule 7: (46, lift 58.6)
##  PC2 <= 7.564459
##  PC3 <= -1.937199
##  PC4 <= -2.765665
##  PC13 > -1.444951
##  PC50 > -0.739172
##  ->  class pos  [0.979]
##
## Rule 8: (29, lift 58.0)
##  PC4 > -2.765665
##  PC5 <= -1.079978
##  PC6 > 1.813945
##  PC7 <= 2.193364
##  PC8 <= 5.190835
```

```
##  PC51 > 0.3047037
##  ->  class pos  [0.968]
##
## Rule 9: (55/1, lift 57.8)
##  PC1 <= -21.68346
##  PC4 <= -2.765665
##  PC9 > -0.7009
##  PC13 > 1.989667
##  PC59 <= 5.833703
##  ->  class pos  [0.965]
##
## Rule 10: (25, lift 57.7)
##  PC5 > -1.079978
##  PC11 <= -2.138421
##  PC13 > 0.575384
##  PC19 <= 0.2403861
##  PC50 > 1.129443
##  PC66 > -2.326112
##  ->  class pos  [0.963]
##
## Rule 11: (23, lift 57.5)
##  PC5 <= -1.079978
##  PC7 <= 2.193364
##  PC19 <= -7.143337
##  PC20 > 1.399353
##  ->  class pos  [0.960]
##
## Rule 12: (55/2, lift 56.7)
##  PC2 <= -2.893974
##  PC4 > -2.765665
##  PC5 <= -1.079978
##  PC7 <= 2.193364
##  PC11 <= 0.09546168
##  PC17 <= 4.183732
##  PC40 > 0.8830376
##  ->  class pos  [0.947]
##
## Rule 13: (36/1, lift 56.7)
##  PC2 <= 0.1437619
##  PC7 > 2.193364
##  PC9 > -0.9003437
##  PC9 <= 1.880038
##  PC15 <= 2.679337
##  PC29 > -0.2164083
##  PC43 > -3.659157
##  PC55 > 0.4701133
##  ->  class pos  [0.947]
##
## Rule 14: (16, lift 56.6)
##  PC4 > -2.765665
##  PC4 <= 6.865797
##  PC7 > 2.193364
##  PC9 > 0.9235685
##  PC9 <= 1.880038
##  PC15 <= 2.679337
##  PC55 <= 0.4701133
##  PC66 > -0.4484272
##  ->  class pos  [0.944]
##
## Rule 15: (42/2, lift 55.8)
##  PC2 <= -2.951965
##  PC5 > -1.079978
##  PC7 <= 2.193364
##  PC11 <= -2.138421
##  PC13 > 0.575384
##  PC43 > -7.199785
##  PC52 <= 2.646349
##  PC57 <= 3.889987
##  PC66 > -2.326112
##  ->  class pos  [0.932]
##
```

```
## Rule 16: (115/7, lift 55.8)
##  PC2 <= -2.256483
##  PC7 > 2.193364
##  PC9 > 1.880038
##  PC33 <= 3.62662
##  ->  class pos  [0.932]
##
## Rule 17: (91/6, lift 55.4)
##  PC4 <= -2.765665
##  PC5 <= -8.086918
##  ->  class pos  [0.925]
##
## Rule 18: (205/15, lift 55.3)
##  PC2 <= -2.893974
##  PC5 <= -1.079978
##  PC11 <= -0.8889982
##  PC17 <= 4.183732
##  ->  class pos  [0.923]
##
## Rule 19: (53/4, lift 54.4)
##  PC4 <= -2.765665
##  PC6 > 3.074957
##  PC56 > -4.314441
##  ->  class pos  [0.909]
##
## Rule 20: (20/1, lift 54.4)
##  PC2 <= 0.1437619
##  PC7 > 2.193364
##  PC9 > -0.9003437
##  PC15 <= 2.679337
##  PC43 > -3.659157
##  PC44 > 0.9629518
##  PC55 > 0.4701133
##  ->  class pos  [0.909]
##
## Rule 21: (31/2, lift 54.4)
##  PC7 > 2.193364
##  PC9 <= 1.880038
##  PC15 > 2.679337
##  PC27 <= -1.283499
##  PC39 > -8.864779
##  ->  class pos  [0.909]
##
## Rule 22: (69/6, lift 54.0)
##  PC2 <= 7.564459
##  PC3 <= -1.937199
##  PC4 <= -2.765665
##  PC13 > -1.444951
##  ->  class pos  [0.901]
##
## Rule 23: (8, lift 53.9)
##  PC2 > -2.893974
##  PC4 > -2.765665
##  PC5 <= -1.079978
##  PC20 > 1.399353
##  PC68 > 3.637683
##  ->  class pos  [0.900]
##
## Rule 24: (26/2, lift 53.5)
##  PC2 > 7.564459
##  PC4 <= -2.765665
##  PC6 <= 3.074957
##  PC19 <= 4.057786
##  PC49 <= -1.689539
##  PC55 > -2.127683
##  ->  class pos  [0.893]
##
## Rule 25: (26/2, lift 53.5)
##  PC2 > 7.564459
##  PC4 <= -14.24129
##  PC13 <= 1.989667
```

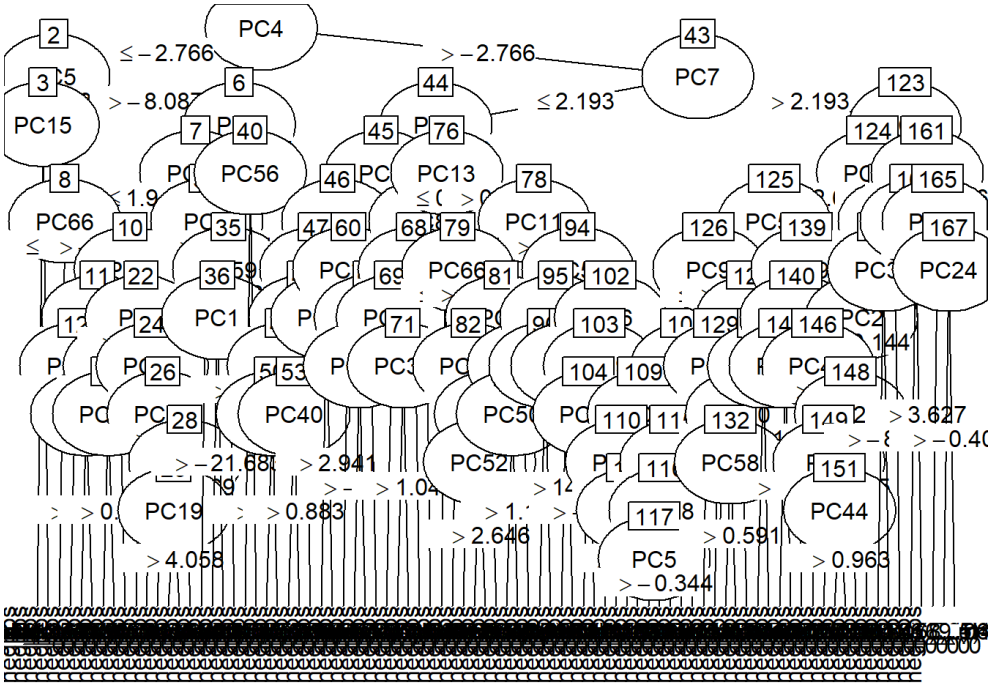
```
##  ->  class pos  [0.893]
##
## Rule 26: (16/1, lift 53.2)
##  PC7 > 2.193364
##  PC9 > 0.9235685
##  PC9 <= 1.880038
##  PC15 <= 2.679337
##  PC29 > 0.129653
##  PC55 <= 0.4701133
##  PC58 > 0.5906942
##  ->  class pos  [0.889]
##
## Rule 27: (69/8, lift 52.3)
##  PC4 <= -2.765665
##  PC66 <= -3.058475
##  ->  class pos  [0.873]
##
## Rule 28: (45/5, lift 52.2)
##  PC2 <= -2.951965
##  PC5 > -1.079978
##  PC7 <= 2.193364
##  PC11 <= -2.138421
##  PC13 > 0.575384
##  PC43 > -7.199785
##  PC57 <= 3.889987
##  PC66 > -2.326112
##  ->  class pos  [0.872]
##
## Rule 29: (273/35, lift 52.0)
##  PC2 <= -2.893974
##  PC5 <= -1.079978
##  PC11 <= 0.09546168
##  PC17 <= 4.183732
##  ->  class pos  [0.869]
##
## Rule 30: (5, lift 51.3)
##  PC4 > -2.765665
##  PC7 > 2.193364
##  PC9 <= -0.9003437
##  PC15 <= 2.679337
##  PC48 > 3.512891
##  PC55 > 0.4701133
##  ->  class pos  [0.857]
##
## Rule 31: (295/42, lift 51.2)
##  PC2 <= -2.893974
##  PC5 <= -1.079978
##  PC11 <= 0.09546168
##  ->  class pos  [0.855]
##
## Rule 32: (68/10, lift 50.5)
##  PC5 > -1.079978
##  PC11 <= -2.138421
##  PC13 > 0.575384
##  PC19 <= 0.2403861
##  PC66 > -2.326112
##  ->  class pos  [0.843]
##
## Rule 33: (62/10, lift 49.6)
##  PC5 <= -1.079978
##  PC6 > 1.813945
##  PC7 <= 2.193364
##  PC8 <= 5.190835
##  ->  class pos  [0.828]
##
## Rule 34: (74/13, lift 48.9)
##  PC4 > -2.765665
##  PC7 > 2.193364
##  PC9 > 0.9235685
##  PC55 <= 0.4701133
##  PC66 > -0.4484272
```

```
##  ->  class pos  [0.816]
##
## Rule 35: (60/12, lift 47.3)
##  PC2 <= 0.1437619
##  PC7 > 2.193364
##  PC9 > -0.9003437
##  PC9 <= 1.880038
##  PC55 > 0.4701133
##  ->  class pos  [0.790]
##
## Rule 36: (57/12, lift 46.7)
##  PC2 > 7.564459
##  PC4 <= -2.765665
##  PC49 <= -1.689539
##  ->  class pos  [0.780]
##
## Rule 37: (45/11, lift 44.6)
##  PC2 > 7.564459
##  PC4 <= -2.765665
##  PC6 <= 3.074957
##  PC19 > -4.099694
##  PC55 <= -2.127683
##  PC66 > -3.058475
##  ->  class pos  [0.745]
##
## Rule 38: (221/57, lift 44.3)
##  PC7 > 2.193364
##  PC9 > 1.880038
##  ->  class pos  [0.740]
##
## Rule 39: (277/93, lift 39.7)
##  PC11 <= -2.138421
##  PC13 > 0.575384
##  PC66 > -2.326112
##  ->  class pos  [0.663]
##
## Default class: neg
##
##
## Evaluation on training data (59646 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      39  454( 0.8%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##   58561    89    (a): class neg
##     365   631    (b): class pos
##
##
##  Attribute usage:
##
##   99.75% PC4
##   99.41% PC5
##   98.99% PC13
##   98.95% PC66
##   98.90% PC6
##   97.96% PC7
##   95.81% PC2
##   94.85% PC20
##   94.21% PC9
##   93.94% PC55
##   93.75% PC15
##   93.58% PC1
##   92.82% PC57
##   92.81% PC51
##   92.76% PC34
```

```
##    0.84% PC11
##    0.46% PC17
##    0.27% PC19
##    0.19% PC33
##    0.15% PC43
##    0.12% PC50
##    0.12% PC3
##    0.10% PC8
##    0.10% PC49
##    0.09% PC40
##    0.09% PC59
##    0.09% PC56
##    0.09% PC29
##    0.07% PC52
##    0.05% PC27
##    0.05% PC39
##    0.03% PC44
##    0.03% PC58
##    0.01% PC68
##    0.01% PC48
##
##
## Time: 12.2 secs
```

Se ha generado un árbol que tiene 39 reglas de decisión y a priori solo tiene un error del 0.8%

```
model <- C50::C5.0(trainx, trainy)
plot(model)
```



Predecimos la clase para los valores de test y calculamos la precisión del árbol.

```
predicted_model <- predict( model, testx, type="class" )
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_model == testy) / length(predicted_model)))
```

```
## [1] "La precisión del árbol es: 97.2438 %"
```

Generamos la matriz de confusión para encontrar mostrar los falsos positivos y los falsos negativos.

```
mat_conf<-table(testy,Predicted=predicted_model)
mat_conf
```

```
##      Predicted
## testy   neg   pos
##   neg 15451   174
##   pos   267   108
```

Calculamos el coste de la predicción, los falsos positivos tienen un coste valorado en 10 unidades, en cambio, los falsos negativos tienen un coste valorado en 500 unidades. Se acumulan 267 falsos negativos. El 71.2% de los casos positivos está mal clasificado. El coste total ha sido de 127300 unidades.

```
coste_total <- mat_conf[1,2]*10 + mat_conf[2,1]*500
coste_total
```

```
## [1] 135240
```

5.3.2 c5.0 con datos imputados con mediana

Se obtiene un modelo con de 50 reglas. La precisión del modelo es 97.39%%. Hay 276 falsos negativos, un número elevado parecido y ligeramente superior al obtenido por el de imputación k-means. El coste total basado en la matriz de confusión es 127300.

```
# Carga de conjunto de datos
train<-train_median_pca
test<-test_median_pca

# Identificación x e y
trainx <- train[,2:66]
trainy <- as.factor(train[,1])
testx <- test[,2:66]
testy <- as.factor(test[,1])

# Modelo
model <- C50::C5.0(trainx, trainy,rules=TRUE )
summary(model)
```

```
##
## Call:
## C5.0.default(x = trainx, y = trainy, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Jan 04 19:16:56 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 59646 cases (66 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (54026/596, lift 1.0)
##  PC3 > -1.360755
##  PC7 > -14.17931
##  ->  class neg  [0.989]
##
## Rule 2: (5568/350, lift 1.0)
##  PC3 <= -1.360755
##  ->  class neg  [0.937]
##
## Rule 3: (103/1, lift 58.7)
##  PC1 <= -6.23485
##  PC3 > -1.360755
##  PC7 <= -2.302865
##  PC8 <= 12.38984
##  PC42 > -0.8579273
##  PC61 <= 0.3777549
##  ->  class pos  [0.981]
##
## Rule 4: (34, lift 58.2)
##  PC1 <= -6.23485
##  PC3 > -1.360755
##  PC7 > -3.87714
##  PC7 <= -2.302865
##  PC13 <= 0.6208433
##  PC51 <= 1.345024
##  PC61 <= 0.3777549
##  ->  class pos  [0.972]
##
## Rule 5: (32, lift 58.1)
##  PC7 <= -2.302865
##  PC8 <= 8.614836
##  PC9 > -4.944446
##  PC16 <= -5.355857
##  PC22 > -4.459424
##  ->  class pos  [0.971]
##
## Rule 6: (27, lift 57.8)
##  PC5 <= 2.660375
##  PC7 > -2.302865
##  PC8 <= -2.26297
##  PC9 > 2.610849
##  PC14 <= 2.339698
##  PC55 > -1.448627
##  ->  class pos  [0.966]
##
## Rule 7: (165/5, lift 57.7)
##  PC1 <= -6.23485
##  PC3 > -1.360755
##  PC7 <= -3.87714
##  PC8 <= 12.38984
##  PC56 > -3.044592
##  ->  class pos  [0.964]
##
## Rule 8: (25, lift 57.7)
##  PC2 <= -5.104641
##  PC5 > 2.660375
##  PC8 <= -2.564271
```

```
##  PC13 > -1.917614
##  PC14 <= 2.339698
##  PC34 > -0.7430986
##  ->  class pos  [0.963]
##
## Rule 9: (51/1, lift 57.6)
##  PC3 > -1.360755
##  PC7 <= -2.302865
##  PC8 <= 12.38984
##  PC23 <= -2.776509
##  ->  class pos  [0.962]
##
## Rule 10: (23, lift 57.5)
##  PC7 > -2.302865
##  PC13 <= -1.917614
##  PC15 > 1.355016
##  PC20 > -1.582655
##  PC26 > -7.197762
##  PC56 > -1.604899
##  PC59 <= -1.510899
##  ->  class pos  [0.960]
##
## Rule 11: (23, lift 57.5)
##  PC1 <= -8.818995
##  PC10 > -0.385254
##  PC14 > 2.339698
##  PC25 <= -0.3383396
##  PC61 <= -1.853034
##  ->  class pos  [0.960]
##
## Rule 12: (20, lift 57.2)
##  PC3 <= -1.360755
##  PC5 <= -5.525272
##  PC7 <= -2.302865
##  PC22 > -4.459424
##  PC36 <= -0.1641926
##  PC51 > -1.780959
##  ->  class pos  [0.955]
##
## Rule 13: (102/4, lift 57.0)
##  PC2 <= -2.165945
##  PC7 > -2.302865
##  PC10 <= 1.177009
##  PC13 <= -1.917614
##  PC15 > 0.1196191
##  ->  class pos  [0.952]
##
## Rule 14: (18, lift 56.9)
##  PC2 > 4.657577
##  PC5 <= 2.660375
##  PC7 > -2.302865
##  PC8 > -2.26297
##  PC18 <= -0.7888103
##  PC43 <= -0.7512645
##  PC62 > -1.085276
##  ->  class pos  [0.950]
##
## Rule 15: (17, lift 56.7)
##  PC1 <= -39.30722
##  PC5 <= 2.660375
##  PC8 <= -2.26297
##  PC13 > -1.917614
##  PC14 <= 2.339698
##  PC47 <= 0.833542
##  ->  class pos  [0.947]
##
## Rule 16: (17, lift 56.7)
##  PC1 <= -8.818995
##  PC3 <= 4.935263
##  PC4 <= 0.2341461
##  PC10 <= -0.385254
```

```
##  PC14 > 2.339698
##  PC15 <= 3.542446
##  PC50 > 1.046265
##  PC54 > -0.6307992
##  PC64 <= -0.3234646
##  ->  class pos  [0.947]
##
## Rule 17: (17, lift 56.7)
##  PC7 > -2.302865
##  PC10 <= -0.385254
##  PC13 > -1.917614
##  PC14 > 2.339698
##  PC46 <= 3.524683
##  PC54 > 2.148435
##  ->  class pos  [0.947]
##
## Rule 18: (34/1, lift 56.6)
##  PC5 > 2.660375
##  PC6 <= -0.1615821
##  PC7 > -2.302865
##  PC14 > -0.155559
##  PC14 <= 2.339698
##  PC29 > 0.1550532
##  PC53 > -1.32849
##  ->  class pos  [0.944]
##
## Rule 19: (15, lift 56.4)
##  PC5 <= -5.525272
##  PC7 <= -2.302865
##  PC26 <= -2.341362
##  PC36 <= -0.1641926
##  ->  class pos  [0.941]
##
## Rule 20: (65/3, lift 56.3)
##  PC4 <= -4.140089
##  PC10 <= -0.385254
##  PC14 > 2.339698
##  PC46 <= 3.524683
##  PC54 > -0.6307992
##  PC64 > -0.3234646
##  ->  class pos  [0.940]
##
## Rule 21: (14, lift 56.1)
##  PC5 <= 2.660375
##  PC7 > -2.302865
##  PC8 <= -2.26297
##  PC9 <= 2.610849
##  PC14 <= 2.339698
##  PC15 > 0.3443652
##  PC51 > 1.882985
##  ->  class pos  [0.938]
##
## Rule 22: (14, lift 56.1)
##  PC7 > -2.302865
##  PC10 <= -0.385254
##  PC13 > -1.917614
##  PC14 > 2.339698
##  PC15 > 3.542446
##  ->  class pos  [0.938]
##
## Rule 23: (75/4, lift 56.0)
##  PC1 <= -8.818995
##  PC10 <= -0.385254
##  PC14 > 2.339698
##  PC46 <= 3.524683
##  PC48 <= 0.5678196
##  PC54 > -0.6307992
##  PC64 > -0.3234646
##  ->  class pos  [0.935]
##
## Rule 24: (28/1, lift 55.9)
```

```
##  PC3 <= -1.360755
##  PC7 <= -2.302865
##  PC22 <= -4.459424
##  PC26 > 0.5799887
##  ->  class pos  [0.933]
##
## Rule 25: (13, lift 55.9)
##  PC3 <= -1.360755
##  PC5 > -5.525272
##  PC6 > -8.106923
##  PC7 <= -2.302865
##  PC17 <= 0.7724722
##  PC30 > 0.7388853
##  PC31 > -1.824009
##  PC55 > -0.8042971
##  PC64 > -0.4945914
##  ->  class pos  [0.933]
##
## Rule 26: (24/1, lift 55.3)
##  PC2 <= -2.098261
##  PC5 <= 2.660375
##  PC6 <= -0.1656561
##  PC7 > -2.302865
##  PC14 <= 2.339698
##  PC20 > -1.082494
##  PC40 > 1.114004
##  ->  class pos  [0.923]
##
## Rule 27: (11, lift 55.3)
##  PC2 > 4.657577
##  PC5 <= 2.660375
##  PC7 > -2.302865
##  PC8 > -2.26297
##  PC13 > -1.917614
##  PC14 <= 2.339698
##  PC58 > -1.97621
##  PC62 <= -1.085276
##  ->  class pos  [0.923]
##
## Rule 28: (10, lift 54.9)
##  PC2 > -2.165945
##  PC7 > -2.302865
##  PC13 <= -1.917614
##  PC15 > 0.1196191
##  PC31 > 2.974623
##  ->  class pos  [0.917]
##
## Rule 29: (21/1, lift 54.7)
##  PC2 > -2.165945
##  PC7 > -2.302865
##  PC13 <= -1.917614
##  PC15 > 0.1196191
##  PC36 > -0.3290075
##  PC56 > -1.604899
##  PC59 > -1.510899
##  ->  class pos  [0.913]
##
## Rule 30: (21/1, lift 54.7)
##  PC5 > 2.660375
##  PC7 > -2.302865
##  PC8 > -2.564271
##  PC14 <= 2.339698
##  PC15 > 0.6601675
##  PC29 > 0.1550532
##  PC58 > -1.088581
##  ->  class pos  [0.913]
##
## Rule 31: (9, lift 54.4)
##  PC2 > 4.657577
##  PC5 <= 2.660375
##  PC7 > -2.302865
```

```
##  PC13 > -1.917614
##  PC14 <= 2.339698
##  PC35 <= 0.8768432
##  PC57 > 2.611253
##  ->  class pos  [0.909]
##
## Rule 32: (9, lift 54.4)
##  PC7 > -2.302865
##  PC13 > -1.917614
##  PC14 > 2.339698
##  PC40 > -8.446465
##  PC49 > 4.15613
##  PC54 <= -0.6307992
##  ->  class pos  [0.909]
##
## Rule 33: (15/1, lift 52.8)
##  PC6 > -8.106923
##  PC7 <= -2.302865
##  PC22 <= -4.459424
##  PC30 > 0.9197772
##  ->  class pos  [0.882]
##
## Rule 34: (56/6, lift 52.7)
##  PC5 <= 2.660375
##  PC7 > -2.302865
##  PC8 <= -2.26297
##  PC9 > 2.610849
##  ->  class pos  [0.879]
##
## Rule 35: (39/4, lift 52.6)
##  PC1 <= -14.48053
##  PC13 <= -1.917614
##  PC15 <= 0.1196191
##  PC18 > 3.389891
##  ->  class pos  [0.878]
##
## Rule 36: (6, lift 52.4)
##  PC2 > 9.363121
##  PC4 > -2.496117
##  PC7 > -2.302865
##  PC10 <= -0.385254
##  PC13 > -1.917614
##  PC14 > 2.339698
##  PC54 <= -0.6307992
##  ->  class pos  [0.875]
##
## Rule 37: (5, lift 51.3)
##  PC3 <= -1.360755
##  PC7 <= -2.302865
##  PC22 <= -4.459424
##  PC26 <= -5.408052
##  ->  class pos  [0.857]
##
## Rule 38: (31/4, lift 50.8)
##  PC1 <= -39.30722
##  PC8 <= -2.26297
##  PC9 <= 2.610849
##  PC13 > -1.917614
##  PC14 <= 2.339698
##  ->  class pos  [0.848]
##
## Rule 39: (161/24, lift 50.7)
##  PC2 <= -2.098261
##  PC6 <= -0.1656561
##  PC40 > 1.114004
##  ->  class pos  [0.847]
##
## Rule 40: (15/2, lift 49.3)
##  PC3 <= -1.360755
##  PC5 > -5.525272
##  PC6 > -8.106923
```

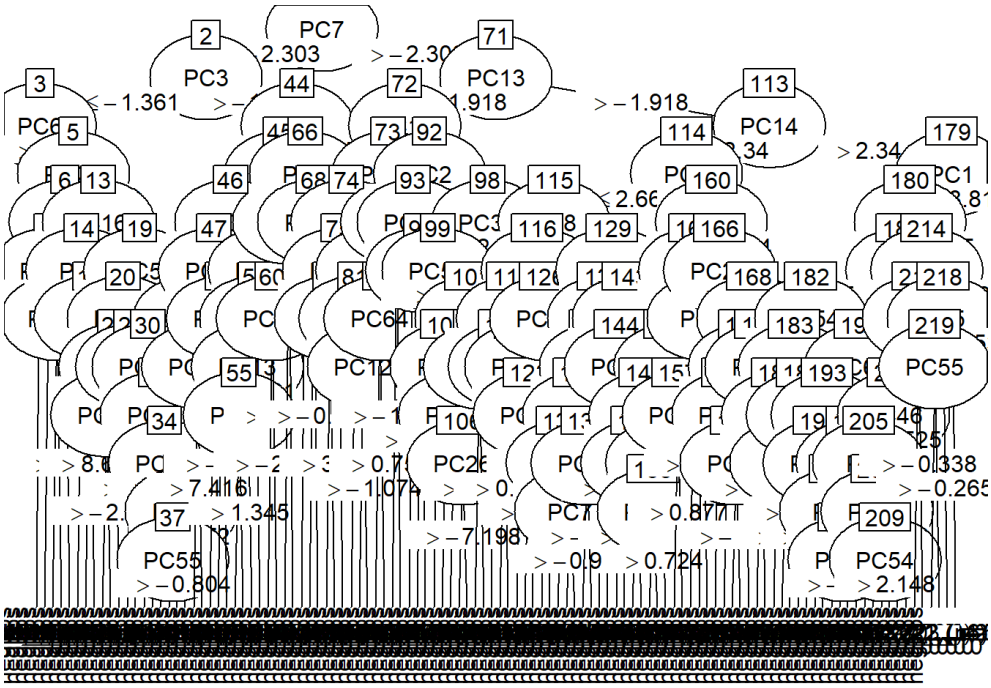
```
##  PC6 <= 8.038406
##  PC7 <= -2.302865
##  PC16 > -5.355857
##  PC17 <= 0.7724722
##  PC30 > 0.7388853
##  PC31 > -1.824009
##  PC64 > -0.4945914
##  ->  class pos  [0.824]
##
## Rule 41: (82/14, lift 49.2)
##  PC6 <= -8.106923
##  PC7 <= -2.302865
##  ->  class pos  [0.821]
##
## Rule 42: (47/8, lift 48.9)
##  PC13 <= -1.917614
##  PC15 <= 0.1196191
##  PC18 > 3.389891
##  ->  class pos  [0.816]
##
## Rule 43: (107/20, lift 48.3)
##  PC7 > -2.302865
##  PC13 <= -1.917614
##  PC15 > 0.1196191
##  PC56 > -1.604899
##  PC59 > -1.510899
##  ->  class pos  [0.807]
##
## Rule 44: (13/2, lift 47.9)
##  PC2 > 4.657577
##  PC5 <= 2.660375
##  PC7 > -2.302865
##  PC8 > -2.26297
##  PC13 > -1.917614
##  PC14 <= 2.339698
##  PC62 <= -1.085276
##  ->  class pos  [0.800]
##
## Rule 45: (47/10, lift 46.4)
##  PC5 > 2.660375
##  PC6 <= -0.1615821
##  PC7 > -2.302865
##  PC14 > -0.155559
##  PC14 <= 2.339698
##  PC29 > 0.1550532
##  ->  class pos  [0.776]
##
## Rule 46: (322/73, lift 46.2)
##  PC3 > -1.360755
##  PC7 <= -2.302865
##  ->  class pos  [0.772]
##
## Rule 47: (88/23, lift 43.9)
##  PC5 > 2.660375
##  PC7 > -2.302865
##  PC14 <= 2.339698
##  PC15 > 0.6601675
##  PC29 > 0.1550532
##  ->  class pos  [0.733]
##
## Rule 48: (35/10, lift 42.1)
##  PC2 > 4.657577
##  PC5 <= 2.660375
##  PC7 > -2.302865
##  PC13 > -1.917614
##  PC57 > 2.611253
##  ->  class pos  [0.703]
##
## Rule 49: (84/25, lift 41.8)
##  PC7 <= -2.302865
##  PC22 <= -4.459424
```

```
##  ->  class pos  [0.698]
##
## Rule 50: (331/111, lift 39.7)
##  PC1 <= -8.818995
##  PC10 <= -0.385254
##  PC14 > 2.339698
##  ->  class pos  [0.664]
##
## Default class: neg
##
##
## Evaluation on training data (59646 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      50  333( 0.6%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##   58601    49    (a): class neg
##     284   712    (b): class pos
##
##
##  Attribute usage:
##
##  100.00% PC3
##   91.10% PC7
##    0.97% PC14
##    0.96% PC1
##    0.73% PC10
##    0.66% PC8
##    0.62% PC13
##    0.61% PC2
##    0.54% PC5
##    0.51% PC15
##    0.49% PC56
##    0.49% PC6
##    0.28% PC40
##    0.23% PC54
##    0.23% PC22
##    0.23% PC64
##    0.22% PC59
##    0.22% PC61
##    0.21% PC9
##    0.19% PC46
##    0.18% PC29
##    0.17% PC42
##    0.15% PC4
##    0.13% PC48
##    0.12% PC26
##    0.11% PC51
##    0.11% PC18
##    0.09% PC36
##    0.09% PC23
##    0.08% PC16
##    0.08% PC20
##    0.07% PC55
##    0.06% PC57
##    0.06% PC53
##    0.05% PC58
##    0.05% PC62
##    0.05% PC30
##    0.04% PC31
##    0.04% PC34
##    0.04% PC25
##    0.03% PC43
##    0.03% PC47
##    0.03% PC50
```

```
##    0.03% PC17
##    0.02% PC35
##    0.02% PC49
##
##
## Time: 12.0 secs
```

Se ha generado un árbol que tiene 50 reglas de decisión y a priori solo tiene un error del 0.6%. La visualización del árbol no es buena, sería difícil utilizarlo como herramienta de decisión.

```
model <- C50::C5.0(trainx, trainy)
plot(model)
```




```
predicted_model <- predict( model, testx, type="class" )
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_model == testy) / length(predicted_model)))
```

```
## [1] "La precisión del árbol es: 97.3063 %"
```

Generamos la matriz de confusión para encontrar mostrar los falsos positivos y los falsos negativos. 276 de los 375 casos positivos en el conjunto de testeo se han clasificado de manera incorrecta, lo que representa el 73.6%.

```
mat_conf<-table(testy,Predicted=predicted_model)
mat_conf
```

```
##       Predicted
## testy   neg   pos
##   neg 15445   180
##   pos   251   124
```

El coste total ha sido de 127300 unidades.

```
coste_total <- mat_conf[1,2]*10 + mat_conf[2,1]*500
coste_total
```

```
## [1] 127300
```

```
trainx<-NULL
trainy<-NULL
testx<-NULL
testy<-NULL
```

5.4 Regresión logística

5.4.1 Regresión logística sobre imputación k-means

5.4.1.1 Regresión logística sobre imputación k-means, sin transformación Cox-Box

Se realiza una regresión logística sin balancear los datos (positivos y negativos) con los datos sin transformación Box-Cox, donde los perdidos han sido imputados por k-means. Se obtiene un modelo con 49 variables significativas. La precisión del modelo es 98.73%. Hay 139 falsos negativos. El 37% de los positivos está mal clasificado. El coste total basado en la matriz de confusión es 70130.

```
train_log<-train_km
test_log<-test_km
train_log$class[train_log$class=="neg"]<-"0"
train_log$class[train_log$class=="pos"]<-"1"
train_log$class<-as.factor(train_log$class)
levels(train_log$class)<- c('neg', 'pos')

summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:DescTools':
##
##     MAE, RMSE
```

```
model_log1 <- glm (class~., data=train_log, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_log1)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial, data = train_log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.2912  -0.0806  -0.0634  -0.0549   3.8196
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.514e+00  1.313e-01 -49.632  < 2e-16 ***
## aa_000       4.091e-06  6.987e-07   5.854 4.79e-09 ***
## ac_000      -2.373e-10  1.152e-10  -2.060 0.039438 *
## ag_000      -2.140e-05  2.842e-05  -0.753 0.451474
## ag_002      -9.523e-07  3.454e-07  -2.757 0.005834 **
## ag_004      -1.207e-06  1.669e-07  -7.232 4.76e-13 ***
## ag_005      -1.103e-06  1.405e-07  -7.856 3.97e-15 ***
## ag_006      -9.834e-07  1.406e-07  -6.996 2.63e-12 ***
## ag_007      -1.155e-06  1.476e-07  -7.828 4.94e-15 ***
## ag_008      -1.072e-06  2.319e-07  -4.621 3.81e-06 ***
## ag_009      -1.180e-06  2.087e-07  -5.652 1.59e-08 ***
## aj_000      -1.379e-06  4.697e-07  -2.937 0.003315 **
## al_000       3.256e-07  1.463e-07   2.225 0.026063 *
## am_0        -1.282e-07  9.767e-08  -1.313 0.189339
## an_000       1.034e-07  2.351e-08   4.399 1.09e-05 ***
## ao_000      -7.382e-08  2.931e-08  -2.519 0.011777 *
## ap_000      -1.595e-07  6.176e-08  -2.582 0.009821 **
## aq_000       4.483e-07  8.373e-08   5.354 8.62e-08 ***
## ar_000       4.852e-03  2.821e-03   1.720 0.085476 .
## as_000      -6.887e-09  1.902e-06  -0.004 0.997111
## at_000       6.241e-07  3.378e-07   1.847 0.064708 .
## au_000       4.755e-06  3.222e-06   1.476 0.139958
## av_000       5.507e-06  4.512e-06   1.221 0.222243
## ax_000      -3.133e-05  2.154e-05  -1.455 0.145797
## ay_000       3.014e-07  2.875e-07   1.048 0.294447
## ay_001       2.466e-07  2.870e-07   0.859 0.390241
## ay_002       4.149e-07  3.061e-07   1.355 0.175280
## ay_003       3.000e-08  3.466e-07   0.087 0.931033
## ay_004       1.194e-07  2.889e-07   0.413 0.679467
## ay_005       1.645e-07  2.800e-07   0.587 0.556871
## ay_006       7.626e-08  2.794e-07   0.273 0.784920
## ay_007       9.962e-08  2.794e-07   0.357 0.721435
## ay_008       2.211e-07  2.795e-07   0.791 0.428864
## ay_009       9.602e-07  3.596e-07   2.670 0.007586 **
## az_000      -3.769e-07  4.878e-07  -0.773 0.439751
## az_001       4.528e-06  1.306e-06   3.466 0.000529 ***
## az_002      -8.397e-07  4.247e-07  -1.977 0.048018 *
## az_003       4.467e-07  1.375e-07   3.249 0.001157 **
## az_004       3.080e-07  1.337e-07   2.304 0.021241 *
## az_005       3.032e-07  1.340e-07   2.264 0.023594 *
## az_006       2.334e-07  1.456e-07   1.603 0.109035
## az_007       1.009e-06  2.180e-07   4.630 3.65e-06 ***
## az_008       1.494e-06  2.194e-06   0.681 0.495950
## az_009      -4.761e-06  3.065e-05  -0.155 0.876552
## ba_002      -1.989e-07  1.066e-07  -1.866 0.062028 .
## ba_003       2.849e-07  1.598e-07   1.783 0.074597 .
## ba_008       6.731e-08  9.327e-08   0.722 0.470490
## ba_009       9.773e-08  1.536e-07   0.636 0.524587
## bb_000       2.454e-08  1.758e-08   1.396 0.162673
## bc_000       5.072e-06  4.827e-06   1.051 0.293371
## bd_000       5.905e-06  3.897e-06   1.515 0.129765
## be_000      -6.346e-06  2.043e-06  -3.105 0.001901 **
## bf_000       5.634e-05  2.520e-05   2.236 0.025384 *
## bg_000      -1.940e-08  3.381e-08  -0.574 0.566132
## bh_000      -4.858e-06  1.006e-06  -4.827 1.38e-06 ***
## bi_000       8.881e-08  6.160e-08   1.442 0.149388
## bj_000       1.445e-07  5.699e-08   2.535 0.011252 *
## bu_000      -3.428e-08  1.897e-08  -1.807 0.070803 .
## bv_000       1.216e-08  1.604e-08   0.758 0.448269
## bx_000       1.520e-08  6.335e-09   2.400 0.016385 *
## ca_000       5.893e-06  1.648e-06   3.576 0.000349 ***
```

```
## cb_000       1.383e-07  1.620e-07   0.854 0.393097
## cc_000      -1.482e-09  7.394e-09  -0.200 0.841162
## ce_000      -1.312e-07  1.623e-07  -0.808 0.418809
## ci_000       7.936e-09  1.407e-08   0.564 0.572880
## cj_000       1.270e-07  2.404e-08   5.283 1.27e-07 ***
## ck_000       6.532e-08  2.232e-08   2.927 0.003425 **
## cl_000       6.789e-06  3.464e-06   1.960 0.050017 .
## cm_000       4.073e-05  1.259e-05   3.234 0.001220 **
## cn_000       3.150e-07  5.805e-07   0.543 0.587378
## cn_001       2.129e-06  4.378e-07   4.863 1.15e-06 ***
## cn_002       3.552e-07  2.630e-07   1.350 0.176862
## cn_003       5.009e-07  2.487e-07   2.014 0.044011 *
## cn_004       5.826e-07  2.406e-07   2.422 0.015454 *
## cn_005       5.088e-07  2.439e-07   2.086 0.036982 *
## cn_006       2.913e-07  2.534e-07   1.149 0.250389
## cn_007       6.669e-07  3.211e-07   2.077 0.037795 *
## cn_008       2.233e-07  3.584e-07   0.623 0.533327
## cn_009       7.548e-07  3.175e-07   2.377 0.017432 *
## cp_000      -1.112e-06  1.528e-06  -0.728 0.466723
## cq_000       9.502e-10  1.287e-08   0.074 0.941127
## cs_000      -1.016e-05  6.851e-06  -1.483 0.138168
## cs_001      -4.860e-06  1.341e-05  -0.362 0.717080
## cs_002      -3.633e-07  2.964e-07  -1.226 0.220276
## cs_003      -3.251e-07  2.974e-07  -1.093 0.274276
## cs_004      -1.454e-07  2.928e-07  -0.497 0.619445
## cs_005      -1.790e-07  2.928e-07  -0.612 0.540809
## cs_006      -1.512e-07  2.937e-07  -0.515 0.606765
## cs_007       3.724e-07  1.221e-06   0.305 0.760392
## cs_008      -1.148e-06  1.323e-05  -0.087 0.930866
## cs_009      -7.651e-05  2.199e-04  -0.348 0.727947
## dd_000      -3.801e-06  2.526e-06  -1.505 0.132313
## de_000       2.695e-06  8.475e-06   0.318 0.750523
## df_000       1.813e-07  6.930e-08   2.616 0.008892 **
## dg_000       5.622e-08  4.731e-08   1.188 0.234650
## dh_000      -6.121e-08  8.421e-08  -0.727 0.467353
## di_000       8.369e-09  3.134e-08   0.267 0.789427
## dj_000      -5.845e-06  7.201e-05  -0.081 0.935311
## dk_000       3.441e-07  1.847e-07   1.864 0.062391 .
## dl_000      -1.965e-05  3.894e-05  -0.505 0.613836
## dm_000       5.679e-07  3.450e-07   1.646 0.099723 .
## dn_000       1.683e-06  1.168e-06   1.441 0.149561
## do_000       1.366e-06  3.789e-07   3.606 0.000311 ***
## dp_000       1.454e-06  1.940e-06   0.749 0.453646
## dq_000      -2.910e-10  3.142e-10  -0.926 0.354359
## dr_000      -2.789e-09  1.380e-08  -0.202 0.839844
## ds_000      -5.771e-08  1.471e-07  -0.392 0.694877
## dt_000       4.136e-06  8.548e-07   4.839 1.31e-06 ***
## du_000      -7.704e-09  2.415e-09  -3.190 0.001422 **
## dv_000      -1.231e-08  9.280e-09  -1.327 0.184574
## dx_000      -6.116e-09  4.703e-09  -1.301 0.193421
## dy_000       1.570e-07  2.892e-07   0.543 0.587246
## dz_000       3.798e-03  1.678e-03   2.263 0.023637 *
## ea_000      -9.899e-05  9.565e-04  -0.103 0.917571
## eb_000      -1.987e-10  4.825e-10  -0.412 0.680498
## ec_00       -1.427e-05  7.752e-06  -1.841 0.065601 .
## ed_000       9.296e-06  8.461e-06   1.099 0.271895
## ee_000       2.814e-07  1.260e-07   2.233 0.025556 *
## ee_001       4.068e-07  1.324e-07   3.072 0.002129 **
## ee_002       4.738e-07  1.555e-07   3.047 0.002311 **
## ee_003      -3.629e-07  2.250e-07  -1.613 0.106806
## ee_004       1.464e-07  1.444e-07   1.014 0.310448
## ee_005       5.050e-07  1.378e-07   3.666 0.000247 ***
## ee_006       2.437e-07  1.347e-07   1.809 0.070400 .
## ee_007       3.830e-08  1.330e-07   0.288 0.773416
## ee_008       5.107e-07  1.531e-07   3.336 0.000851 ***
## ee_009      -1.212e-06  5.955e-07  -2.036 0.041777 *
## ef_000      -1.091e-02  1.243e-02  -0.877 0.380248
## eg_000      -1.517e-03  3.322e-03  -0.457 0.647928
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10127.4  on 59645  degrees of freedom
## Residual deviance:  3302.4  on 59517  degrees of freedom
## AIC: 3560.4
##
## Number of Fisher Scoring iterations: 18
```

```
# Mostrar las variables significativas (p<0.05)
sig_coeff<-data.frame(summary(model_log1)$coef[summary(model_log1)$coef[,4] <= .05,c(4)])
names(sig_coeff)<-c("p_value")
dim(sig_coeff)[1]
```

```
## [1] 49
```

```
rownames(sig_coeff)
```

```
##  [1] "(Intercept)" "aa_000"      "ac_000"      "ag_002"      "ag_004"
##  [6] "ag_005"      "ag_006"      "ag_007"      "ag_008"      "ag_009"
## [11] "aj_000"      "al_000"      "an_000"      "ao_000"      "ap_000"
## [16] "aq_000"      "ay_009"      "az_001"      "az_002"      "az_003"
## [21] "az_004"      "az_005"      "az_007"      "be_000"      "bf_000"
## [26] "bh_000"      "bj_000"      "bx_000"      "ca_000"      "cj_000"
## [31] "ck_000"      "cm_000"      "cn_001"      "cn_003"      "cn_004"
## [36] "cn_005"      "cn_007"      "cn_009"      "df_000"      "do_000"
## [41] "dt_000"      "du_000"      "dz_000"      "ee_000"      "ee_001"
## [46] "ee_002"      "ee_005"      "ee_008"      "ee_009"
```

```
## Predicción de valores
predict <- predict(model_log1, test_log, type = 'response')

## Matriz de confusión
cm<-table(test_log$class, predict > 0.5)
cm
```

```
##
##       FALSE  TRUE
##   neg 15562    63
##   pos   139   236
```

```
precision<-(cm[1,1]+cm[2,2])/sum(cm);precision
```

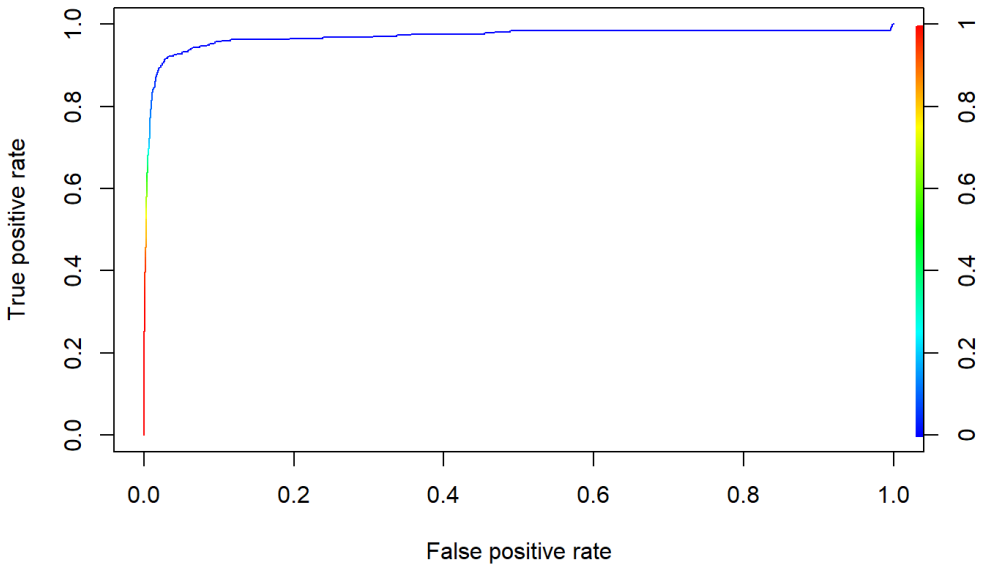
```
## [1] 0.987375
```

```
## Coste Total
coste_total_log <- cm[1,2]*10 + cm[2,1]*500
coste_total_log
```

```
## [1] 70130
```

Se dibuja la curva ROC. La curva ROC es una representación gráfica del rendimiento del clasificador que muestra la distribución de las fracciones de verdaderos positivos y de falsos positivos. El AUC o área de la cuadrícula que queda bajo la curva ROC es 96.94%.

```
#ROCR Curve
library(ROCR)
ROCRpred <- prediction(predict, test_log$class)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```




```
# AUC value
auc_ROCR <- performance(ROCRpred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]];auc_ROCR
```

```
## [1] 0.9694908
```

5.4.1.2 Regresión logística sobre imputación k-means, **con transformación Cox-Box**

Se replica el caso anterior, utilizando ahora el conjunto de datos con imputación de perdidos por k-means y **transformación de datos Box-Cox logística**. Se obtiene un modelo con 42 variables significativas. La precisión del modelo es 98.75%. Hay 132 falsos negativos. El 35.2% de los casos positivos están mal clasificados. El coste total basado en la matriz de confusión es 66670. El indicador AIC (cuanto menor mejor) es 3034.

```
train_log<-train_bc_km
test_log<-test_bc_km
train_log$class[train_log$class=="neg"]<-"0"
train_log$class[train_log$class=="pos"]<-"1"
train_log$class<-as.factor(train_log$class)
levels(train_log$class)<- c('neg', 'pos')

summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
prop.table(table(train_log$class))
```

```
##
##        neg        pos
## 0.98330148 0.01669852
```

```
table(train_log$class)
```

```
##
##   neg   pos
## 58650   996
```

```
library(caret)

model_log1 <- glm (class~., data=train_log, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_log1)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial, data = train_log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8605  -0.0392  -0.0171  -0.0007   6.4656
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.056e+01  1.756e+00 -17.403  < 2e-16 ***
## aa_000       1.997e-01  9.954e-02   2.006 0.044834 *
## ac_000       5.209e-03  1.431e-02   0.364 0.715753
## ag_000       2.414e-02  5.739e-02   0.421 0.674030
## ag_002       1.343e-02  2.949e-02   0.456 0.648747
## ag_004      -2.465e-01  8.390e-02  -2.937 0.003309 **
## ag_005      -8.766e-02  1.412e-01  -0.621 0.534666
## ag_006      -1.243e-01  1.189e-01  -1.045 0.296075
## ag_007      -1.099e-01  6.891e-02  -1.594 0.110831
## ag_008       2.113e-02  3.791e-02   0.557 0.577264
## ag_009      -5.679e-02  2.047e-02  -2.775 0.005525 **
## aj_000       1.114e-02  1.930e-02   0.577 0.563727
## al_000       2.002e-01  6.175e-02   3.243 0.001185 **
## am_0        -1.084e-01  5.769e-02  -1.879 0.060231 .
## an_000       3.031e-01  2.744e-01   1.105 0.269199
## ao_000      -7.077e-01  1.739e-01  -4.070 4.71e-05 ***
## ap_000      -2.740e-01  1.289e-01  -2.126 0.033494 *
## aq_000       8.954e-01  2.214e-01   4.044 5.25e-05 ***
## ar_000       3.542e-02  5.611e-02   0.631 0.527931
## as_000       1.537e-03  7.157e-02   0.021 0.982871
## at_000      -2.446e-02  1.820e-02  -1.344 0.178952
## au_000       1.326e-01  5.775e-02   2.296 0.021702 *
## av_000       2.592e-02  2.019e-02   1.284 0.199143
## ax_000      -3.718e-02  2.325e-02  -1.599 0.109823
## ay_000       2.542e-01  5.580e-02   4.555 5.24e-06 ***
## ay_001      -2.030e-01  8.833e-02  -2.298 0.021548 *
## ay_002       2.585e-01  7.706e-02   3.354 0.000796 ***
## ay_003      -1.096e-01  5.165e-02  -2.122 0.033822 *
## ay_004       6.264e-02  2.319e-02   2.701 0.006917 **
## ay_005       9.063e-03  1.884e-02   0.481 0.630550
## ay_006      -1.455e-01  2.009e-02  -7.242 4.42e-13 ***
## ay_007      -1.782e-02  4.413e-02  -0.404 0.686377
## ay_008       7.926e-02  3.148e-02   2.518 0.011806 *
## ay_009       1.668e-01  2.798e-02   5.962 2.49e-09 ***
## az_000       7.401e-02  1.278e-01   0.579 0.562661
## az_001       4.732e-01  1.416e-01   3.341 0.000834 ***
## az_002      -1.287e-01  1.234e-01  -1.043 0.296999
## az_003       2.026e-01  7.287e-02   2.780 0.005440 **
## az_004      -7.515e-02  5.391e-02  -1.394 0.163378
## az_005       1.339e-01  9.825e-02   1.362 0.173065
## az_006      -9.727e-03  2.334e-02  -0.417 0.676870
## az_007       9.609e-04  2.993e-02   0.032 0.974389
## az_008      -2.661e-02  3.740e-02  -0.711 0.476832
## az_009       2.979e-02  5.922e-02   0.503 0.614927
## ba_002      -7.002e-01  1.826e-01  -3.834 0.000126 ***
## ba_003       5.210e-01  1.910e-01   2.728 0.006377 **
## ba_008      -6.485e-02  2.639e-02  -2.458 0.013987 *
## ba_009       3.452e-02  2.508e-02   1.376 0.168753
## bb_000       8.263e-01  4.242e-01   1.948 0.051403 .
## bc_000       5.282e-02  2.260e-02   2.337 0.019441 *
## bd_000       9.871e-02  2.909e-02   3.394 0.000690 ***
## be_000      -3.300e-02  3.064e-02  -1.077 0.281392
## bf_000       1.256e-03  2.270e-02   0.055 0.955865
## bg_000      -1.065e+00  1.687e-01  -6.316 2.69e-10 ***
## bh_000      -7.543e-02  2.672e-01  -0.282 0.777682
## bi_000      -3.400e-01  1.234e-01  -2.754 0.005879 **
## bj_000       4.554e-01  1.929e-01   2.360 0.018258 *
## bu_000      -4.463e-02  1.527e-01  -0.292 0.770050
## bv_000      -7.129e-02  3.019e-01  -0.236 0.813334
## bx_000       2.339e-02  5.656e-02   0.414 0.679183
## ca_000      -2.529e-03  3.778e-02  -0.067 0.946625
```

```
## cb_000      -6.807e-02  4.923e-02  -1.383 0.166751
## cc_000       3.605e-02  3.888e-02   0.927 0.353744
## ce_000      -2.364e-03  1.126e-02  -0.210 0.833732
## ci_000       1.403e+00  1.951e-01   7.190 6.48e-13 ***
## cj_000       2.496e-02  1.277e-02   1.954 0.050692 .
## ck_000       1.009e-02  1.539e-01   0.066 0.947730
## cl_000      -3.433e-02  2.283e-02  -1.504 0.132625
## cm_000      -1.821e-02  1.925e-02  -0.946 0.344187
## cn_000       1.825e-01  2.741e-02   6.657 2.79e-11 ***
## cn_001      -5.654e-02  2.726e-02  -2.074 0.038087 *
## cn_002       1.382e-01  4.625e-02   2.988 0.002805 **
## cn_003       1.245e-01  1.278e-01   0.974 0.330063
## cn_004       4.944e-01  2.402e-01   2.059 0.039504 *
## cn_005       3.279e-01  2.075e-01   1.581 0.113970
## cn_006      -2.269e-01  1.416e-01  -1.602 0.109061
## cn_007      -1.168e-01  1.561e-01  -0.749 0.454009
## cn_008       1.628e-01  1.273e-01   1.279 0.200962
## cn_009       9.633e-02  4.945e-02   1.948 0.051417 .
## cp_000      -3.935e-02  1.799e-02  -2.187 0.028745 *
## cq_000      -5.734e-02  1.463e-01  -0.392 0.695180
## cs_000      -3.871e-01  2.444e-01  -1.584 0.113259
## cs_001      -3.319e-01  2.012e-01  -1.649 0.099053 .
## cs_002      -1.379e-01  6.177e-02  -2.233 0.025579 *
## cs_003       1.743e-01  1.379e-01   1.264 0.206255
## cs_004       2.386e-01  1.137e-01   2.099 0.035796 *
## cs_005      -1.329e-01  1.370e-01  -0.970 0.331940
## cs_006      -3.952e-02  7.795e-02  -0.507 0.612143
## cs_007      -1.476e-02  4.086e-02  -0.361 0.717931
## cs_008      -3.489e-02  4.053e-02  -0.861 0.389345
## cs_009       2.219e-02  1.054e-01   0.210 0.833310
## dd_000       1.739e-05  3.493e-02   0.000 0.999603
## de_000      -2.538e-02  3.601e-02  -0.705 0.480828
## df_000       4.712e-02  2.557e-02   1.843 0.065388 .
## dg_000       2.943e-02  2.370e-02   1.241 0.214438
## dh_000      -6.473e-03  1.807e-02  -0.358 0.720119
## di_000      -1.150e-02  1.161e-02  -0.990 0.322093
## dj_000       1.408e-02  1.105e-01   0.127 0.898604
## dk_000       5.659e-02  4.638e-02   1.220 0.222449
## dl_000      -4.429e-01  1.689e-01  -2.622 0.008740 **
## dm_000       3.181e-01  9.811e-02   3.242 0.001188 **
## dn_000       3.479e-01  2.141e-01   1.625 0.104185
## do_000       3.192e-02  2.849e-02   1.120 0.262537
## dp_000      -1.668e-02  3.181e-02  -0.524 0.600097
## dq_000      -1.377e-02  9.707e-03  -1.419 0.155896
## dr_000      -2.021e-02  1.162e-02  -1.739 0.081987 .
## ds_000       1.311e-02  2.707e-02   0.484 0.628242
## dt_000       4.007e-02  3.314e-02   1.209 0.226673
## du_000      -2.129e-02  1.745e-02  -1.220 0.222596
## dv_000      -2.517e-02  1.884e-02  -1.336 0.181466
## dx_000      -1.988e-02  1.017e-02  -1.955 0.050539 .
## dy_000       8.919e-03  1.403e-02   0.636 0.525010
## dz_000       1.552e-01  1.508e-01   1.029 0.303334
## ea_000      -1.510e-01  1.150e-01  -1.313 0.189200
## eb_000      -5.939e-03  9.746e-03  -0.609 0.542301
## ec_00       -4.393e-02  2.395e-02  -1.834 0.066674 .
## ed_000      -2.635e-02  2.680e-02  -0.983 0.325539
## ee_000      -9.434e-02  1.792e-01  -0.526 0.598636
## ee_001      -2.066e-01  2.112e-01  -0.978 0.328066
## ee_002       1.145e-01  1.734e-01   0.660 0.509017
## ee_003      -4.979e-02  1.249e-01  -0.399 0.690105
## ee_004      -4.352e-01  1.387e-01  -3.137 0.001706 **
## ee_005       4.965e-01  1.080e-01   4.597 4.29e-06 ***
## ee_006       2.100e-01  7.755e-02   2.708 0.006773 **
## ee_007      -4.895e-01  6.221e-02  -7.869 3.58e-15 ***
## ee_008       2.348e-01  5.481e-02   4.284 1.84e-05 ***
## ee_009       4.922e-03  2.604e-02   0.189 0.850052
## ef_000       1.303e-01  2.191e-01   0.595 0.552134
## eg_000      -6.029e-02  1.539e-01  -0.392 0.695275
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10127  on 59645  degrees of freedom
## Residual deviance:  2776  on 59517  degrees of freedom
## AIC: 3034
##
## Number of Fisher Scoring iterations: 12
```

```
# Mostrar las variables significativas (p<0.05)
sig_coeff<-data.frame(summary(model_log1)$coef[summary(model_log1)$coef[,4] <= .05,c(4)])
names(sig_coeff)<-c("p_value")
dim(sig_coeff)[1]
```

```
## [1] 42
```

```
rownames(sig_coeff)
```

```
##  [1] "(Intercept)" "aa_000"      "ag_004"      "ag_009"      "al_000"
##  [6] "ao_000"      "ap_000"      "aq_000"      "au_000"      "ay_000"
## [11] "ay_001"      "ay_002"      "ay_003"      "ay_004"      "ay_006"
## [16] "ay_008"      "ay_009"      "az_001"      "az_003"      "ba_002"
## [21] "ba_003"      "ba_008"      "bc_000"      "bd_000"      "bg_000"
## [26] "bi_000"      "bj_000"      "ci_000"      "cn_000"      "cn_001"
## [31] "cn_002"      "cn_004"      "cp_000"      "cs_002"      "cs_004"
## [36] "dl_000"      "dm_000"      "ee_004"      "ee_005"      "ee_006"
## [41] "ee_007"      "ee_008"
```

```
## Predicción de valores
predict <- predict(model_log1, test_log, type = 'response')

## Matriz de confusión
cm<-table(test_log$class, predict > 0.5)
cm
```

```
##
##       FALSE  TRUE
##   neg 15558    67
##   pos   132   243
```

```
precision<-(cm[1,1]+cm[2,2])/sum(cm);precision
```

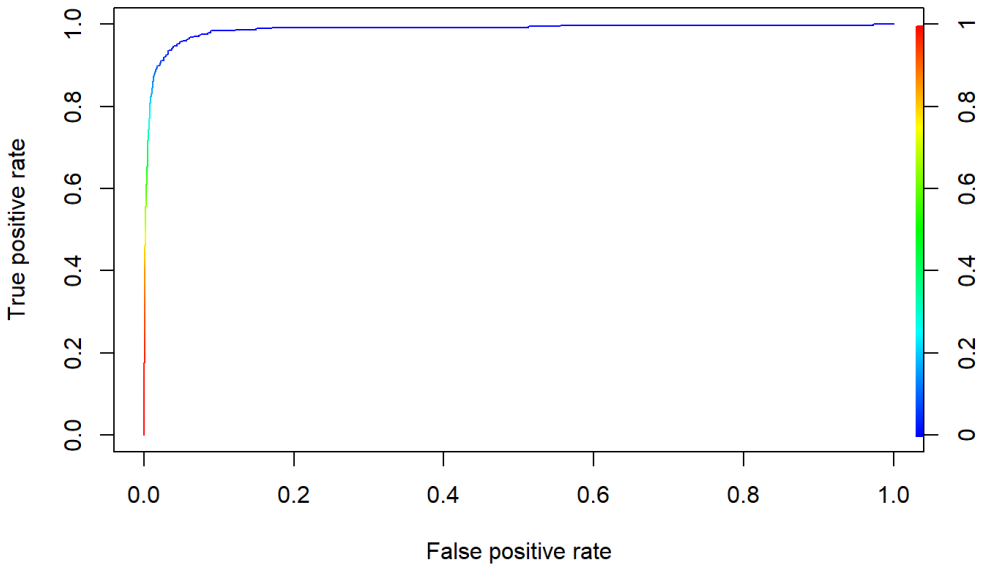
```
## [1] 0.9875625
```

```
## Coste Total
coste_total_log <- cm[1,2]*10 + cm[2,1]*500
coste_total_log
```

```
## [1] 66670
```

Se dibuja la curva ROC. La curva ROC es una representación gráfica del rendimiento del clasificador que muestra la distribución de las fracciones de verdaderos positivos y de falsos positivos. El AUC o área de la cuadrícula que queda bajo la curva ROC es 98.66%.

```
#ROCR Curve
library(ROCR)
ROCRpred <- prediction(predict, test_log$class)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```




```
# AUC value
auc_ROCR <- performance(ROCRpred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]];auc_ROCR
```

```
## [1] 0.9866993
```

5.4.1.3 Regresión logística, imputación k-means, datos transformados Box-Cox y con datos balanceados mediante SMOTE

Se continúa este caso con datos transformados Box-Cot con imputación k-means. Balanceado de datos con SMOTE [[6]]. Se utilizan como parámetros perc.over=300 y perc.under=150. El perc.over amplifica el número de clases de la clase minoritaria, mientras que perc.under aminora la mayoritaria. De esta forma, se balancea la proporción y se obtienen 4482 negativos y 3984 positivos.

Se obtiene un modelo con 48 variables significativas. La precisión del modelo es 95.63%. Hay 32 falsos negativos. Solo el 8.5% de los casos positivos están mal clasificados, una mejora muy significativa. El coste total basado en la matriz de confusión es 22660. El indicador AIC (cuanto menor mejor) es 2051.

```
library(DMwR)
```

```
## Loading required package: grid
```

```
train_log<-train_bc_km
test_log<-test_bc_km
train_log$class[train_log$class=="neg"]<-"0"
train_log$class[train_log$class=="pos"]<-"1"
train_log$class<-as.factor(train_log$class)
levels(train_log$class)<- c('neg', 'pos')

summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
prop.table(table(train_log$class))
```

```
##
##        neg        pos
## 0.98330148 0.01669852
```

```
## Smote : Synthetic Minority Oversampling Technique To Handle Class Imbalancy In Binary Classification
train_balanced <- SMOTE(class ~., train_log, perc.over = 300, k = 5, perc.under = 150)
summary(train_balanced$class)
```

```
##  neg  pos
## 4482 3984
```

```
library(caret)
ctrl<-glm.control(epsilon = 1e-8, maxit =100, trace = FALSE)
model_log2 <- glm (class~., data=train_balanced, family = binomial,control=ctrl)
summary(model_log2)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial, data = train_balanced,
##     control = ctrl)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0121  -0.0773  -0.0020   0.0735   4.3580
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.675e+01  2.250e+00 -11.886  < 2e-16 ***
## aa_000       1.500e-01  1.277e-01   1.174 0.240232
## ac_000       4.503e-02  1.700e-02   2.649 0.008062 **
## ag_000       1.608e-01  1.087e-01   1.480 0.138852
## ag_002      -1.504e-02  5.123e-02  -0.294 0.769063
## ag_004      -3.860e-01  1.061e-01  -3.640 0.000273 ***
## ag_005      -8.749e-01  1.394e-01  -6.275 3.50e-10 ***
## ag_006       5.767e-01  1.926e-01   2.994 0.002750 **
## ag_007      -1.515e-01  9.141e-02  -1.657 0.097505 .
## ag_008      -2.041e-02  5.475e-02  -0.373 0.709292
## ag_009      -1.448e-01  3.021e-02  -4.793 1.65e-06 ***
## aj_000       7.580e-02  3.089e-02   2.453 0.014149 *
## al_000       2.069e-01  9.678e-02   2.138 0.032533 *
## am_0        -1.900e-01  9.193e-02  -2.067 0.038723 *
## an_000      -1.212e+00  3.982e-01  -3.045 0.002329 **
## ao_000      -9.935e-01  3.029e-01  -3.280 0.001039 **
## ap_000      -2.737e-01  1.663e-01  -1.646 0.099813 .
## aq_000       5.431e-01  3.068e-01   1.770 0.076679 .
## ar_000      -1.223e-01  1.081e-01  -1.131 0.257880
## as_000      -1.930e-01  1.380e-01  -1.398 0.161997
## at_000       4.134e-02  2.971e-02   1.391 0.164126
## au_000       9.230e-02  9.673e-02   0.954 0.339961
## av_000      -6.437e-02  3.854e-02  -1.670 0.094857 .
## ax_000      -7.661e-02  4.306e-02  -1.779 0.075265 .
## ay_000       5.174e-01  1.024e-01   5.050 4.41e-07 ***
## ay_001      -4.638e-01  1.954e-01  -2.373 0.017638 *
## ay_002       5.601e-01  1.742e-01   3.216 0.001300 **
## ay_003      -2.146e-01  8.690e-02  -2.469 0.013556 *
## ay_004       1.117e-02  4.181e-02   0.267 0.789327
## ay_005       1.091e-02  2.624e-02   0.416 0.677631
## ay_006      -1.410e-01  3.230e-02  -4.364 1.28e-05 ***
## ay_007      -1.869e-01  5.351e-02  -3.492 0.000479 ***
## ay_008       6.085e-02  3.496e-02   1.741 0.081735 .
## ay_009       2.900e-01  7.659e-02   3.787 0.000153 ***
## az_000       4.864e-01  2.019e-01   2.410 0.015971 *
## az_001       1.901e-01  1.922e-01   0.989 0.322625
## az_002      -2.051e-02  1.697e-01  -0.121 0.903787
## az_003       3.044e-01  1.000e-01   3.044 0.002337 **
## az_004      -2.193e-01  7.726e-02  -2.838 0.004539 **
## az_005       3.348e-01  1.453e-01   2.305 0.021162 *
## az_006      -7.013e-04  3.186e-02  -0.022 0.982440
## az_007      -9.693e-02  4.201e-02  -2.307 0.021036 *
## az_008      -5.511e-02  5.818e-02  -0.947 0.343525
## az_009       3.363e-02  1.060e-01   0.317 0.751014
## ba_002      -8.030e-01  3.270e-01  -2.455 0.014076 *
## ba_003       4.519e-01  2.617e-01   1.727 0.084191 .
## ba_008      -5.404e-02  3.224e-02  -1.676 0.093741 .
## ba_009       9.494e-02  3.440e-02   2.760 0.005779 **
## bb_000       1.233e+00  6.138e-01   2.009 0.044504 *
## bc_000       1.742e-01  3.497e-02   4.982 6.28e-07 ***
## bd_000       3.072e-02  4.845e-02   0.634 0.526064
## be_000      -1.415e-01  5.747e-02  -2.462 0.013834 *
## bf_000      -7.336e-02  3.904e-02  -1.879 0.060241 .
## bg_000      -1.167e+00  3.008e-01  -3.881 0.000104 ***
## bh_000      -5.064e-02  3.830e-01  -0.132 0.894825
## bi_000       7.272e-02  1.792e-01   0.406 0.684901
## bj_000       6.771e-01  2.572e-01   2.632 0.008489 **
## bu_000       1.526e-01  4.863e-01   0.314 0.753638
## bv_000       5.102e-01  4.686e-01   1.089 0.276240
## bx_000       3.024e-01  8.980e-02   3.368 0.000757 ***
```

```
## ca_000      -4.015e-02  6.396e-02  -0.628 0.530205
## cb_000      -2.641e-01  7.165e-02  -3.686 0.000228 ***
## cc_000       1.928e-02  5.733e-02   0.336 0.736656
## ce_000       3.065e-02  2.059e-02   1.488 0.136656
## ci_000       1.768e+00  2.680e-01   6.598 4.16e-11 ***
## cj_000      -8.486e-03  2.562e-02  -0.331 0.740482
## ck_000      -4.226e-01  2.229e-01  -1.896 0.057959 .
## cl_000      -1.042e-01  4.134e-02  -2.521 0.011694 *
## cm_000       1.586e-02  3.184e-02   0.498 0.618420
## cn_000       3.623e-01  5.382e-02   6.731 1.69e-11 ***
## cn_001      -1.204e-01  4.080e-02  -2.952 0.003160 **
## cn_002       3.482e-01  5.373e-02   6.481 9.08e-11 ***
## cn_003       4.291e-01  1.518e-01   2.827 0.004701 **
## cn_004       1.654e+00  3.129e-01   5.287 1.24e-07 ***
## cn_005      -6.899e-02  2.886e-01  -0.239 0.811070
## cn_006       4.303e-01  2.368e-01   1.817 0.069206 .
## cn_007       1.409e-01  2.064e-01   0.683 0.494804
## cn_008      -7.363e-01  1.631e-01  -4.515 6.32e-06 ***
## cn_009       4.403e-01  6.860e-02   6.418 1.38e-10 ***
## cp_000       1.492e-02  3.183e-02   0.469 0.639287
## cq_000       1.482e-01  2.139e-01   0.693 0.488369
## cs_000      -6.476e-01  3.287e-01  -1.970 0.048790 *
## cs_001       5.548e-03  2.798e-01   0.020 0.984181
## cs_002      -2.301e-01  8.083e-02  -2.847 0.004416 **
## cs_003       4.012e-01  2.002e-01   2.004 0.045027 *
## cs_004      -2.394e-01  1.914e-01  -1.251 0.211012
## cs_005      -6.193e-01  2.111e-01  -2.933 0.003356 **
## cs_006      -4.336e-01  1.335e-01  -3.248 0.001164 **
## cs_007       1.976e-01  7.144e-02   2.765 0.005686 **
## cs_008      -1.055e-01  5.661e-02  -1.864 0.062363 .
## cs_009       6.654e-02  2.286e-01   0.291 0.770995
## dd_000      -6.422e-02  6.817e-02  -0.942 0.346213
## de_000       2.058e-01  6.321e-02   3.256 0.001132 **
## df_000       1.997e-01  8.370e-02   2.387 0.017006 *
## dg_000      -2.529e-02  5.698e-02  -0.444 0.657172
## dh_000      -8.371e-02  3.563e-02  -2.349 0.018808 *
## di_000       8.738e-02  2.709e-02   3.226 0.001255 **
## dj_000      -1.521e-01  2.900e-01  -0.524 0.600014
## dk_000      -1.061e-01  9.380e-02  -1.131 0.257918
## dl_000      -8.851e-01  2.205e-01  -4.014 5.98e-05 ***
## dm_000       5.446e-01  1.338e-01   4.069 4.71e-05 ***
## dn_000       9.129e-01  3.570e-01   2.557 0.010552 *
## do_000       1.070e-02  5.885e-02   0.182 0.855718
## dp_000       1.913e-02  5.656e-02   0.338 0.735229
## dq_000      -2.991e-04  1.833e-02  -0.016 0.986985
## dr_000      -1.223e-02  2.207e-02  -0.554 0.579361
## ds_000       6.114e-02  5.574e-02   1.097 0.272686
## dt_000       1.911e-03  6.717e-02   0.028 0.977303
## du_000      -4.571e-02  3.771e-02  -1.212 0.225497
## dv_000      -4.190e-03  3.963e-02  -0.106 0.915807
## dx_000      -1.560e-02  1.771e-02  -0.881 0.378405
## dy_000      -3.359e-02  2.584e-02  -1.300 0.193573
## dz_000       2.813e-01  3.030e-01   0.928 0.353189
## ea_000      -5.224e-01  2.011e-01  -2.598 0.009384 **
## eb_000      -2.875e-02  1.493e-02  -1.925 0.054172 .
## ec_00       -2.788e-02  4.652e-02  -0.599 0.548984
## ed_000      -1.126e-01  5.432e-02  -2.073 0.038213 *
## ee_000      -4.564e-02  2.919e-01  -0.156 0.875759
## ee_001      -4.508e-02  3.056e-01  -0.147 0.882740
## ee_002      -4.419e-01  2.880e-01  -1.534 0.124987
## ee_003      -2.379e-01  2.499e-01  -0.952 0.341023
## ee_004      -2.070e-01  2.622e-01  -0.790 0.429733
## ee_005       2.620e-01  1.815e-01   1.444 0.148772
## ee_006       3.617e-01  1.140e-01   3.174 0.001505 **
## ee_007      -4.359e-01  9.204e-02  -4.737 2.17e-06 ***
## ee_008       2.314e-01  7.754e-02   2.985 0.002836 **
## ee_009      -5.143e-02  3.571e-02  -1.440 0.149772
## ef_000       6.386e-02  3.865e-01   0.165 0.868764
## eg_000      -1.244e-01  3.241e-01  -0.384 0.701053
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11707.1  on 8465  degrees of freedom
## Residual deviance:  1720.1  on 8337  degrees of freedom
## AIC: 1978.1
##
## Number of Fisher Scoring iterations: 9
```

```
# Mostrar las variables significativas (p<0.05)
sig_coeff2<-data.frame(summary(model_log2)$coef[summary(model_log2)$coef[,4] <= .05,c(4)])
names(sig_coeff2)<-c("p_value")
dim(sig_coeff2)[1]
```

```
## [1] 59
```

```
rownames(sig_coeff2)
```

```
##  [1] "(Intercept)" "ac_000"      "ag_004"      "ag_005"      "ag_006"
##  [6] "ag_009"      "aj_000"      "al_000"      "am_0"        "an_000"
## [11] "ao_000"      "ay_000"      "ay_001"      "ay_002"      "ay_003"
## [16] "ay_006"      "ay_007"      "ay_009"      "az_000"      "az_003"
## [21] "az_004"      "az_005"      "az_007"      "ba_002"      "ba_009"
## [26] "bb_000"      "bc_000"      "be_000"      "bg_000"      "bj_000"
## [31] "bx_000"      "cb_000"      "ci_000"      "cl_000"      "cn_000"
## [36] "cn_001"      "cn_002"      "cn_003"      "cn_004"      "cn_008"
## [41] "cn_009"      "cs_000"      "cs_002"      "cs_003"      "cs_005"
## [46] "cs_006"      "cs_007"      "de_000"      "df_000"      "dh_000"
## [51] "di_000"      "dl_000"      "dm_000"      "dn_000"      "ea_000"
## [56] "ed_000"      "ee_006"      "ee_007"      "ee_008"
```

```
## Predicción de valores
predict <- predict(model_log2, test_log, type = 'response')

## Matriz de confusión
cm<-table(test_log$class, predict > 0.5)
cm
```

```
##
##       FALSE  TRUE
##   neg 14968   657
##   pos    28   347
```

```
precision<-(cm[1,1]+cm[2,2])/sum(cm);precision
```

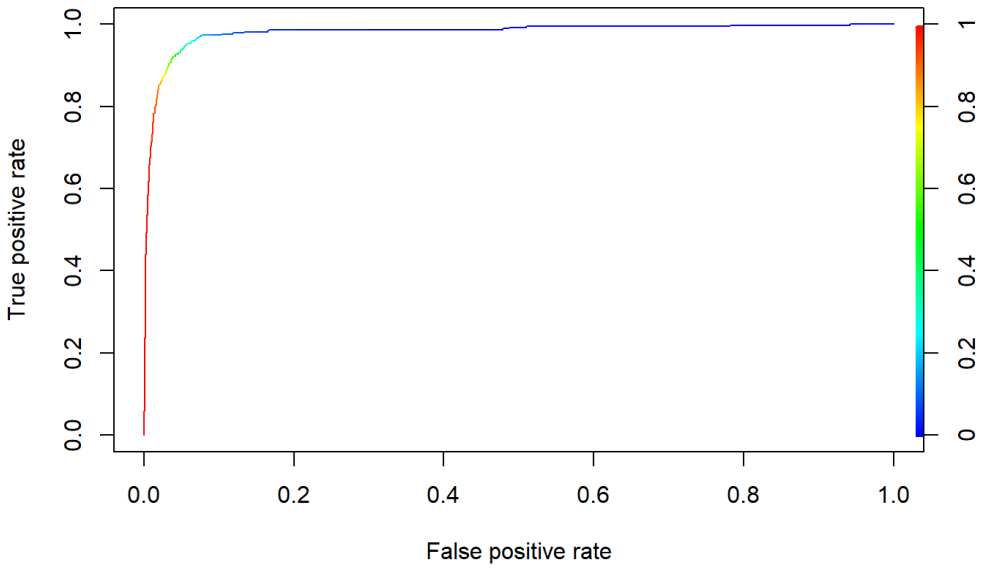
```
## [1] 0.9571875
```

```
## Coste Total
coste_total_log <- cm[1,2]*10 + cm[2,1]*500
coste_total_log
```

```
## [1] 20570
```

Se dibuja la curva ROC. La curva ROC es una representación gráfica del rendimiento del clasificador que muestra la distribución de las fracciones de verdaderos positivos y de falsos positivos. El AUC o área de la cuadrícula que queda bajo la curva ROC es 97.84%.

```
#ROCR Curve
library(ROCR)
ROCRpred <- prediction(predict, test_log$class)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```




```
# AUC value
auc_ROCR <- performance(ROCRpred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]];auc_ROCR
```

```
## [1] 0.9804853
```

5.4.1.4 Regresión logística, imputación k-means, datos transformados Box-Cox y con datos balanceados mediante pesos

Se continúa este caso con datos transformados Box-Cot con imputación k-means. Balanceado de datos con Weights, de forma que el peso de todos los negativos es 0.5 y todos los positivos es 0.5 (suma total de pesos igual a 1). El hecho de incrementat el peso de los positivos hace que el modelo resultante sea más sensible al error de falso negativo. En este caso ninguna de las variables predictoras resulta ser significativa, si bien el modelo se ajusta bien a los datos de testeo y el coste total es muy bajo. Sin embargo, que ninguna variable predictora resulte ser significativa es un mal resultado.

Si en vez de utilizar los datos transformados Box-Cox se utilizan los datos sin transformar, al modelo le cuesta converger (número de iteraciones máximo impuesto 100) y resulta que las 129 variables son significativas, hecho que tampoco es deseble.

Se muestra la ejecución con datos transformados Box-Cox.Se obtiene un modelo con 0 variables significativas. La precisión del modelo es 95.63%. Hay 23 falsos negativos. El coste total basado en la matriz de confusión es 17770. El indicador AIC (cuanto menor mejor) es 258.

```
train_log<-train_bc_km
test_log<-test_bc_km
train_log$class[train_log$class=="neg"]<-"0"
train_log$class[train_log$class=="pos"]<-"1"
train_log$class<-as.factor(train_log$class)
levels(train_log$class)<- c('neg', 'pos')

summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
# Create model weights (they sum to one)

model_weights <- ifelse(train_log$class == "neg",
                        (1/table(train_log$class)[1])*0.5 ,
                        (1/table(train_log$class)[2])*0.5 )
summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
sum(model_weights)
```

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

```
library(caret)
ctrl<-glm.control(epsilon = 1e-8, maxit =100, trace = FALSE)
model_log3 <- glm (class~.,data=train_log, weights=model_weights,family = binomial,control=ctrl)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_log3)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial, data = train_log,
##     weights = model_weights, control = ctrl)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.015836  -0.000721  -0.000314  -0.000082   0.087865
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.172e+01  1.690e+02  -0.129    0.898
## aa_000       4.743e-02  8.814e+00   0.005    0.996
## ac_000       4.491e-02  1.310e+00   0.034    0.973
## ag_000       1.712e-01  9.252e+00   0.019    0.985
## ag_002      -4.163e-02  3.812e+00  -0.011    0.991
## ag_004      -1.461e-01  8.243e+00  -0.018    0.986
## ag_005      -5.825e-01  1.074e+01  -0.054    0.957
## ag_006       2.335e-01  1.386e+01   0.017    0.987
## ag_007      -1.273e-01  6.872e+00  -0.019    0.985
## ag_008       4.072e-02  3.756e+00   0.011    0.991
## ag_009      -1.198e-01  2.375e+00  -0.050    0.960
## aj_000       7.900e-02  2.376e+00   0.033    0.973
## al_000       1.516e-01  6.345e+00   0.024    0.981
## am_0        -1.221e-01  6.072e+00  -0.020    0.984
## an_000      -6.339e-01  2.680e+01  -0.024    0.981
## ao_000      -8.288e-01  2.118e+01  -0.039    0.969
## ap_000      -2.409e-01  1.124e+01  -0.021    0.983
## aq_000       8.547e-01  1.901e+01   0.045    0.964
## ar_000      -4.647e-02  8.268e+00  -0.006    0.996
## as_000      -1.000e-01  1.220e+01  -0.008    0.993
## at_000       4.770e-02  2.345e+00   0.020    0.984
## au_000       1.809e-01  1.016e+01   0.018    0.986
## av_000      -7.900e-03  2.822e+00  -0.003    0.998
## ax_000      -9.799e-02  3.227e+00  -0.030    0.976
## ay_000       3.255e-01  6.994e+00   0.047    0.963
## ay_001      -2.237e-01  1.111e+01  -0.020    0.984
## ay_002       3.012e-01  1.101e+01   0.027    0.978
## ay_003      -1.241e-01  7.324e+00  -0.017    0.986
## ay_004       2.481e-02  3.154e+00   0.008    0.994
## ay_005       3.681e-03  2.049e+00   0.002    0.999
## ay_006      -7.464e-02  2.457e+00  -0.030    0.976
## ay_007      -1.002e-01  4.289e+00  -0.023    0.981
## ay_008       4.271e-02  2.827e+00   0.015    0.988
## ay_009       1.945e-01  5.022e+00   0.039    0.969
## az_000       5.361e-01  1.510e+01   0.036    0.972
## az_001       6.489e-02  1.512e+01   0.004    0.997
## az_002       1.203e-01  1.404e+01   0.009    0.993
## az_003       1.425e-01  7.519e+00   0.019    0.985
## az_004      -1.061e-01  5.780e+00  -0.018    0.985
## az_005       2.461e-01  1.078e+01   0.023    0.982
## az_006      -2.188e-02  2.474e+00  -0.009    0.993
## az_007      -4.608e-02  3.213e+00  -0.014    0.989
## az_008      -2.681e-02  4.644e+00  -0.006    0.995
## az_009      -4.141e-04  8.564e+00   0.000    1.000
## ba_002      -5.039e-01  2.196e+01  -0.023    0.982
## ba_003       9.925e-02  2.056e+01   0.005    0.996
## ba_008      -8.995e-02  2.578e+00  -0.035    0.972
## ba_009       1.016e-01  2.709e+00   0.038    0.970
## bb_000       1.114e+00  3.316e+01   0.034    0.973
## bc_000       1.436e-01  2.766e+00   0.052    0.959
## bd_000       3.183e-02  3.677e+00   0.009    0.993
## be_000      -7.366e-02  4.314e+00  -0.017    0.986
## bf_000      -1.311e-02  3.053e+00  -0.004    0.997
## bg_000      -1.029e+00  2.230e+01  -0.046    0.963
## bh_000      -2.616e-01  2.581e+01  -0.010    0.992
## bi_000      -1.864e-01  1.416e+01  -0.013    0.989
## bj_000       4.377e-01  2.055e+01   0.021    0.983
## bu_000       3.475e-01  3.487e+01   0.010    0.992
## bv_000      -9.589e-03  3.736e+01   0.000    1.000
## bx_000       1.921e-01  7.017e+00   0.027    0.978
```

```
## ca_000      -6.952e-03  4.708e+00  -0.001    0.999
## cb_000      -2.547e-01  5.707e+00  -0.045    0.964
## cc_000       5.592e-02  5.289e+00   0.011    0.992
## ce_000      -1.148e-02  1.543e+00  -0.007    0.994
## ci_000       1.526e+00  2.368e+01   0.064    0.949
## cj_000       1.029e-02  1.911e+00   0.005    0.996
## ck_000      -3.286e-01  2.018e+01  -0.016    0.987
## cl_000      -5.769e-02  3.117e+00  -0.019    0.985
## cm_000       1.812e-02  2.471e+00   0.007    0.994
## cn_000       2.831e-01  4.023e+00   0.070    0.944
## cn_001      -9.410e-02  3.060e+00  -0.031    0.975
## cn_002       2.055e-01  3.949e+00   0.052    0.958
## cn_003       2.359e-01  1.113e+01   0.021    0.983
## cn_004       1.042e+00  2.168e+01   0.048    0.962
## cn_005       1.020e-01  2.032e+01   0.005    0.996
## cn_006      -3.525e-02  1.539e+01  -0.002    0.998
## cn_007       3.168e-01  1.538e+01   0.021    0.984
## cn_008      -5.503e-01  1.168e+01  -0.047    0.962
## cn_009       3.070e-01  4.959e+00   0.062    0.951
## cp_000      -5.413e-03  2.358e+00  -0.002    0.998
## cq_000       8.127e-03  1.747e+01   0.000    1.000
## cs_000      -3.167e-01  2.153e+01  -0.015    0.988
## cs_001      -1.817e-01  1.979e+01  -0.009    0.993
## cs_002      -1.693e-01  6.247e+00  -0.027    0.978
## cs_003       1.896e-02  1.513e+01   0.001    0.999
## cs_004      -4.143e-02  1.372e+01  -0.003    0.998
## cs_005      -6.613e-01  1.509e+01  -0.044    0.965
## cs_006      -1.367e-02  9.328e+00  -0.001    0.999
## cs_007      -6.929e-03  5.443e+00  -0.001    0.999
## cs_008      -1.139e-01  4.281e+00  -0.027    0.979
## cs_009       2.227e-02  1.252e+01   0.002    0.999
## dd_000       5.166e-02  4.929e+00   0.010    0.992
## de_000       6.443e-02  5.129e+00   0.013    0.990
## df_000       9.062e-02  5.858e+00   0.015    0.988
## dg_000       5.579e-02  4.565e+00   0.012    0.990
## dh_000      -9.377e-03  2.713e+00  -0.003    0.997
## di_000      -2.799e-03  1.824e+00  -0.002    0.999
## dj_000      -1.496e-01  1.749e+01  -0.009    0.993
## dk_000      -5.051e-03  7.132e+00  -0.001    0.999
## dl_000      -9.771e-01  2.950e+01  -0.033    0.974
## dm_000       6.033e-01  1.752e+01   0.034    0.973
## dn_000       8.094e-01  2.612e+01   0.031    0.975
## do_000      -3.377e-02  4.200e+00  -0.008    0.994
## dp_000      -2.811e-02  4.642e+00  -0.006    0.995
## dq_000      -1.251e-02  1.393e+00  -0.009    0.993
## dr_000      -2.600e-02  1.647e+00  -0.016    0.987
## ds_000       8.609e-02  3.753e+00   0.023    0.982
## dt_000       8.097e-02  4.409e+00   0.018    0.985
## du_000      -3.536e-02  2.807e+00  -0.013    0.990
## dv_000      -6.460e-02  3.108e+00  -0.021    0.983
## dx_000      -1.009e-03  1.339e+00  -0.001    0.999
## dy_000      -1.237e-02  1.940e+00  -0.006    0.995
## dz_000       3.967e-01  2.795e+01   0.014    0.989
## ea_000      -4.994e-01  1.792e+01  -0.028    0.978
## eb_000      -2.461e-02  1.167e+00  -0.021    0.983
## ec_00       -4.689e-02  3.505e+00  -0.013    0.989
## ed_000      -3.375e-02  3.800e+00  -0.009    0.993
## ee_000       1.575e-01  2.407e+01   0.007    0.995
## ee_001      -2.118e-01  2.234e+01  -0.009    0.992
## ee_002      -8.313e-02  1.966e+01  -0.004    0.997
## ee_003      -6.399e-02  1.728e+01  -0.004    0.997
## ee_004      -1.833e-01  1.660e+01  -0.011    0.991
## ee_005       1.214e-01  1.232e+01   0.010    0.992
## ee_006       4.260e-01  8.658e+00   0.049    0.961
## ee_007      -3.486e-01  6.872e+00  -0.051    0.960
## ee_008       1.554e-01  5.994e+00   0.026    0.979
## ee_009      -2.544e-02  2.779e+00  -0.009    0.993
## ef_000       2.723e-01  3.545e+01   0.008    0.994
## eg_000       9.954e-02  2.293e+01   0.004    0.997
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 1.38629  on 59645  degrees of freedom
## Residual deviance: 0.23868  on 59517  degrees of freedom
## AIC: 258
##
## Number of Fisher Scoring iterations: 9
```

En la matriz de confusión, los falsos negativos se reducen a 23, aunque hayan aumentado los falsos positivos el coste total se reduce a 17770.

```
# Mostrar las variables significativas (p<0.05)
sig_coeff3<-data.frame(summary(model_log3)$coef[summary(model_log3)$coef[,4] <= .05,c(4)])
names(sig_coeff3)<-c("p_value")
dim(sig_coeff3)[1]
```

```
## [1] 0
```

```
rownames(sig_coeff3)
```

```
## character(0)
```

```
## Predicción de valores
predict <- predict(model_log3, test_log, type = 'response')

## Matriz de confusión
cm<-table(test_log$class, predict > 0.5)
cm
```

```
##
##       FALSE  TRUE
##   neg 14998   627
##   pos    23   352
```

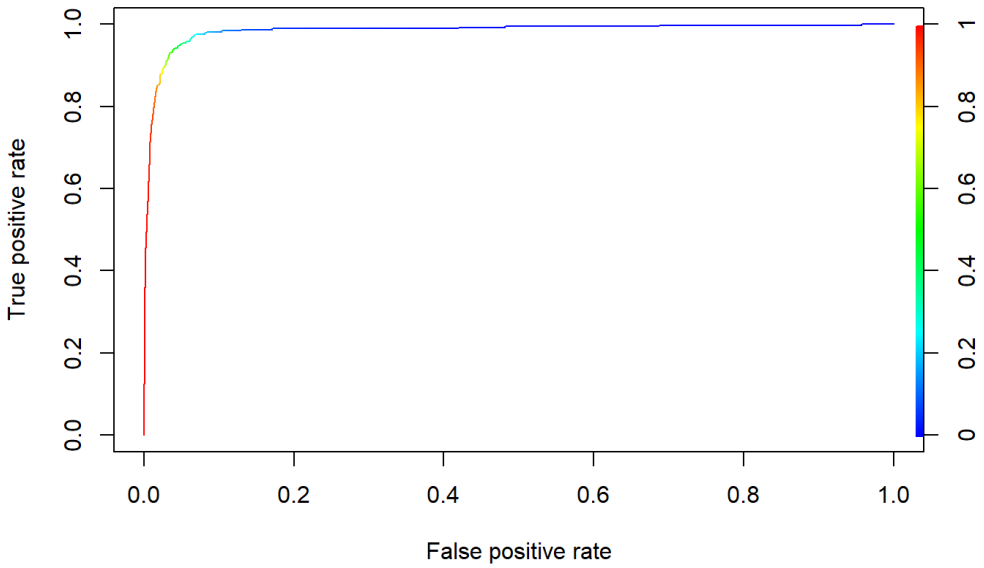
```
precision<-(cm[1,1]+cm[2,2])/sum(cm);precision
```

```
## [1] 0.959375
```

```
## Coste Total
coste_total_log <- cm[1,2]*10 + cm[2,1]*500
coste_total_log
```

```
## [1] 17770
```

```
#ROCR Curve
library(ROCR)
ROCRpred <- prediction(predict, test_log$class)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```




```
# AUC value
auc_ROCR <- performance(ROCRpred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]];auc_ROCR
```

```
## [1] 0.9835682
```

5.4.2 Regresión logística sobre imputación mediana

Se replica ahora el análisis con datos perdidos imputados por la mediana.

5.4.2.1 Regresión logística, imputación mediana, datos no transformados

Se realiza una regresión logística sin balancear los datos (positivos y negativos) con los datos sin transformación Box-Cox, donde los perdidos han sido imputados por le mediana. Se obtiene un modelo con 37 variables significativas. La precisión del modelo es 98.73%. Hay 133 falsos negativos. El 35% de los positivos está mal clasificado. El coste total basado en la matriz de confusión es 66960.

```
train_log<-train_median
test_log<-test_median
train_log$class[train_log$class=="neg"]<-"0"
train_log$class[train_log$class=="pos"]<-"1"
train_log$class<-as.factor(train_log$class)
levels(train_log$class)<- c('neg', 'pos')

summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
library(caret)

model_log1 <- glm (class~., data=train_log, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_log1)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial, data = train_log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6759  -0.0816  -0.0621  -0.0563   4.0354
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.440e+00  1.333e-01 -48.325  < 2e-16 ***
## aa_000       4.993e-06  7.023e-07   7.109 1.17e-12 ***
## ac_000      -3.699e-10  1.383e-10  -2.675 0.007474 **
## ag_000      -8.488e-06  2.254e-05  -0.377 0.706492
## ag_002      -8.751e-07  3.609e-07  -2.425 0.015315 *
## ag_004      -1.274e-06  2.338e-07  -5.451 5.02e-08 ***
## ag_005      -1.166e-06  1.957e-07  -5.959 2.55e-09 ***
## ag_006      -1.099e-06  1.995e-07  -5.506 3.67e-08 ***
## ag_007      -1.242e-06  2.030e-07  -6.117 9.52e-10 ***
## ag_008      -1.265e-06  3.111e-07  -4.067 4.77e-05 ***
## ag_009      -1.297e-06  2.791e-07  -4.647 3.37e-06 ***
## aj_000      -1.681e-06  4.870e-07  -3.452 0.000557 ***
## al_000       2.005e-07  3.248e-07   0.617 0.537026
## am_0         2.560e-09  2.194e-07   0.012 0.990691
## an_000       1.525e-07  7.973e-08   1.913 0.055719 .
## ao_000      -1.821e-07  8.895e-08  -2.048 0.040592 *
## ap_000      -2.605e-07  2.040e-07  -1.277 0.201592
## aq_000       5.820e-07  1.356e-07   4.291 1.78e-05 ***
## ar_000       3.312e-03  3.440e-03   0.963 0.335745
## as_000       8.016e-07  1.777e-06   0.451 0.651875
## at_000       9.333e-07  3.836e-07   2.433 0.014962 *
## au_000       3.046e-06  3.046e-06   1.000 0.317276
## av_000       1.978e-05  7.411e-06   2.669 0.007605 **
## ax_000      -1.308e-04  5.024e-05  -2.603 0.009253 **
## ay_000      -8.091e-05  7.990e-05  -1.013 0.311222
## ay_001      -8.099e-05  7.990e-05  -1.014 0.310737
## ay_002      -8.075e-05  7.990e-05  -1.011 0.312150
## ay_003      -8.128e-05  7.990e-05  -1.017 0.309003
## ay_004      -8.116e-05  7.990e-05  -1.016 0.309739
## ay_005      -8.104e-05  7.990e-05  -1.014 0.310418
## ay_006      -8.115e-05  7.990e-05  -1.016 0.309780
## ay_007      -8.112e-05  7.990e-05  -1.015 0.309961
## ay_008      -8.100e-05  7.990e-05  -1.014 0.310688
## ay_009      -7.990e-05  7.990e-05  -1.000 0.317285
## az_000       1.146e-04  1.042e-04   1.099 0.271589
## az_001       1.181e-04  1.042e-04   1.134 0.256926
## az_002       1.142e-04  1.042e-04   1.096 0.273281
## az_003       1.152e-04  1.042e-04   1.105 0.268991
## az_004       1.150e-04  1.042e-04   1.104 0.269581
## az_005       1.150e-04  1.042e-04   1.104 0.269647
## az_006       1.149e-04  1.042e-04   1.103 0.270118
## az_007       1.157e-04  1.042e-04   1.111 0.266703
## az_008       1.160e-04  1.042e-04   1.113 0.265555
## az_009       1.118e-04  1.122e-04   0.996 0.319228
## ba_002      -1.246e-07  1.138e-07  -1.095 0.273351
## ba_003       4.061e-07  1.658e-07   2.450 0.014297 *
## ba_008       1.498e-07  8.170e-08   1.833 0.066798 .
## ba_009       2.144e-08  1.745e-07   0.123 0.902196
## bb_000       4.075e-09  4.103e-08   0.099 0.920887
## bc_000       1.363e-05  1.057e-05   1.290 0.197161
## bd_000       8.376e-06  6.717e-06   1.247 0.212435
## be_000      -1.970e-07  3.146e-06  -0.063 0.950072
## bf_000       7.152e-05  3.287e-05   2.176 0.029558 *
## bg_000      -3.025e-08  4.790e-08  -0.632 0.527657
## bh_000      -5.189e-06  1.233e-06  -4.210 2.56e-05 ***
## bi_000       1.924e-07  1.820e-07   1.057 0.290584
## bj_000       1.540e-07  1.949e-07   0.790 0.429278
## bu_000       5.308e-02  2.194e-02   2.420 0.015538 *
## bv_000       4.811e-03  4.106e-03   1.172 0.241304
## bx_000      -5.250e-08  4.771e-08  -1.101 0.271082
## ca_000       6.079e-06  1.936e-06   3.140 0.001689 **
```

```
## cb_000       4.227e-08  1.732e-07   0.244 0.807232
## cc_000       7.009e-08  5.249e-08   1.335 0.181778
## ce_000      -3.887e-08  2.830e-07  -0.137 0.890751
## ci_000      -8.920e-10  1.476e-08  -0.060 0.951816
## cj_000       9.372e-08  3.052e-08   3.071 0.002133 **
## ck_000       9.934e-08  2.476e-08   4.012 6.03e-05 ***
## cl_000       1.106e-05  3.953e-06   2.798 0.005147 **
## cm_000       5.256e-05  2.665e-05   1.972 0.048603 *
## cn_000      -3.651e-06  4.181e-05  -0.087 0.930417
## cn_001      -2.685e-06  4.181e-05  -0.064 0.948793
## cn_002      -3.860e-06  4.181e-05  -0.092 0.926438
## cn_003      -3.810e-06  4.181e-05  -0.091 0.927381
## cn_004      -3.664e-06  4.181e-05  -0.088 0.930158
## cn_005      -3.712e-06  4.181e-05  -0.089 0.929240
## cn_006      -4.052e-06  4.181e-05  -0.097 0.922784
## cn_007      -3.394e-06  4.181e-05  -0.081 0.935298
## cn_008      -4.256e-06  4.181e-05  -0.102 0.918919
## cn_009      -3.472e-06  4.181e-05  -0.083 0.933803
## cp_000      -2.534e-06  2.226e-06  -1.138 0.254965
## cq_000      -5.789e-02  2.248e-02  -2.575 0.010011 *
## cs_000       1.264e-04  1.041e-04   1.215 0.224476
## cs_001       1.714e-04  1.142e-04   1.501 0.133366
## cs_002       1.411e-04  1.042e-04   1.354 0.175600
## cs_003       1.411e-04  1.042e-04   1.355 0.175402
## cs_004       1.414e-04  1.042e-04   1.357 0.174762
## cs_005       1.413e-04  1.042e-04   1.357 0.174897
## cs_006       1.414e-04  1.042e-04   1.357 0.174806
## cs_007       1.426e-04  1.041e-04   1.369 0.170915
## cs_008       1.389e-04  1.062e-04   1.308 0.190930
## cs_009      -2.871e-06  2.969e-04  -0.010 0.992284
## dd_000      -8.650e-06  3.854e-06  -2.245 0.024795 *
## de_000      -2.008e-06  1.065e-05  -0.188 0.850529
## df_000       4.576e-08  1.125e-07   0.407 0.684120
## dg_000       4.545e-07  1.666e-07   2.728 0.006366 **
## dh_000       1.137e-10  1.022e-06   0.000 0.999911
## di_000       1.724e-07  6.217e-08   2.774 0.005541 **
## dj_000      -1.595e-03  2.316e-03  -0.689 0.491007
## dk_000       8.206e-07  4.129e-07   1.987 0.046910 *
## dl_000      -1.288e-05  7.854e-05  -0.164 0.869744
## dm_000       3.818e-07  2.553e-06   0.150 0.881144
## dn_000      -5.038e-07  1.712e-06  -0.294 0.768601
## do_000       5.050e-06  1.567e-06   3.223 0.001267 **
## dp_000      -1.105e-05  7.909e-06  -1.397 0.162421
## dq_000      -3.832e-11  4.884e-10  -0.078 0.937460
## dr_000      -1.185e-08  2.231e-08  -0.531 0.595432
## ds_000      -8.516e-07  4.436e-07  -1.920 0.054874 .
## dt_000       1.203e-05  2.561e-06   4.699 2.62e-06 ***
## du_000      -7.610e-09  4.608e-09  -1.652 0.098615 .
## dv_000      -4.575e-08  1.702e-08  -2.689 0.007176 **
## dx_000      -1.882e-08  9.249e-09  -2.035 0.041875 *
## dy_000      -9.685e-08  5.311e-07  -0.182 0.855295
## dz_000       5.936e-03  3.348e-03   1.773 0.076226 .
## ea_000      -8.437e-04  1.686e-03  -0.500 0.616846
## eb_000      -1.377e-09  5.833e-10  -2.362 0.018201 *
## ec_00       -8.502e-05  2.005e-05  -4.240 2.23e-05 ***
## ed_000       7.129e-05  2.001e-05   3.562 0.000368 ***
## ee_000      -1.704e-04  1.205e-04  -1.413 0.157547
## ee_001      -1.702e-04  1.205e-04  -1.412 0.157836
## ee_002      -1.702e-04  1.205e-04  -1.412 0.158011
## ee_003      -1.711e-04  1.205e-04  -1.420 0.155714
## ee_004      -1.705e-04  1.205e-04  -1.415 0.157200
## ee_005      -1.702e-04  1.205e-04  -1.412 0.157968
## ee_006      -1.704e-04  1.205e-04  -1.414 0.157353
## ee_007      -1.707e-04  1.205e-04  -1.416 0.156850
## ee_008      -1.702e-04  1.205e-04  -1.412 0.157995
## ee_009      -1.719e-04  1.205e-04  -1.426 0.153953
## ef_000      -1.101e-02  2.191e-02  -0.503 0.615190
## eg_000       1.779e-03  4.282e-03   0.415 0.677831
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10127.4  on 59645  degrees of freedom
## Residual deviance:  3216.6  on 59517  degrees of freedom
## AIC: 3474.6
##
## Number of Fisher Scoring iterations: 18
```

```
# Mostrar las variables significativas (p<0.05)
sig_coeff<-data.frame(summary(model_log1)$coef[summary(model_log1)$coef[,4] <= .05,c(4)])
names(sig_coeff)<-c("p_value")
dim(sig_coeff)[1]
```

```
## [1] 37
```

```
rownames(sig_coeff)
```

```
##  [1] "(Intercept)" "aa_000"      "ac_000"      "ag_002"      "ag_004"
##  [6] "ag_005"      "ag_006"      "ag_007"      "ag_008"      "ag_009"
## [11] "aj_000"      "ao_000"      "aq_000"      "at_000"      "av_000"
## [16] "ax_000"      "ba_003"      "bf_000"      "bh_000"      "bu_000"
## [21] "ca_000"      "cj_000"      "ck_000"      "cl_000"      "cm_000"
## [26] "cq_000"      "dd_000"      "dg_000"      "di_000"      "dk_000"
## [31] "do_000"      "dt_000"      "dv_000"      "dx_000"      "eb_000"
## [36] "ec_00"       "ed_000"
```

```
## Predicción de valores
predict <- predict(model_log1, test_log, type = 'response')

## Matriz de confusión
cm<-table(test_log$class, predict > 0.5)
cm
```

```
##
##       FALSE  TRUE
##   neg 15579    46
##   pos   133   242
```

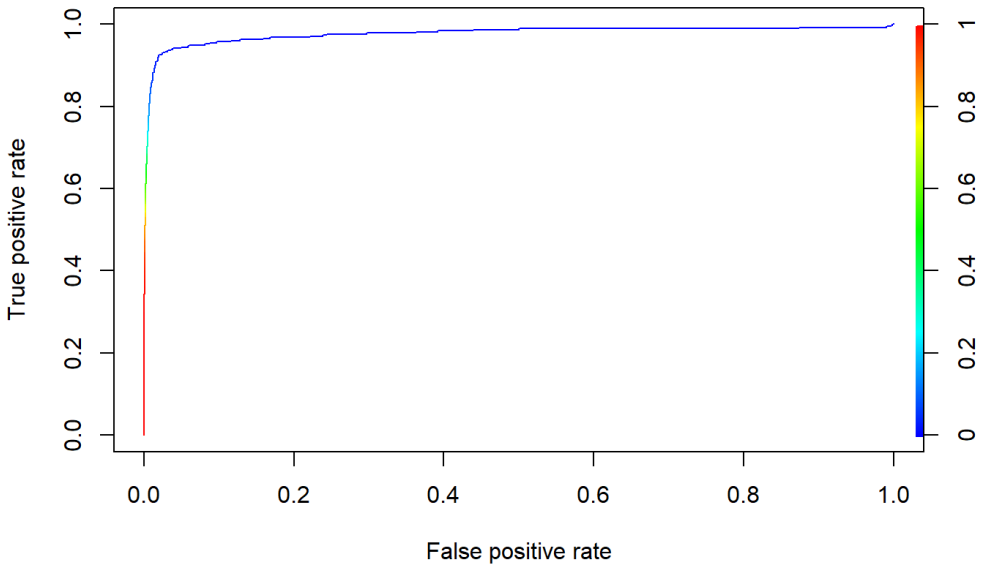
```
precision<-(cm[1,1]+cm[2,2])/sum(cm);precision
```

```
## [1] 0.9888125
```

```
## Coste Total
coste_total_log <- cm[1,2]*10 + cm[2,1]*500
coste_total_log
```

```
## [1] 66960
```

```
#ROCR Curve
library(ROCR)
ROCRpred <- prediction(predict, test_log$class)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```




```
# AUC value
auc_ROCR <- performance(ROCRpred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]];auc_ROCR
```

```
## [1] 0.9765557
```

5.4.2.2 Regresión logística, imputación mediana, datos transformados Box-Cox

Se replica el caso anterior, utilizando ahora el conjunto de datos con imputación de perdidos por mediana y transformación de datos Box-Cox logística. Se obtiene un modelo con 37 variables significativas. La precisión del modelo es 98.93%. Hay 127 falsos negativos. El 33.8% de los positivos está mal clasificado. El coste total basado en la matriz de confusión es 63940.

```
train_log<-train_bc_median
test_log<-test_bc_median
train_log$class[train_log$class=="neg"]<-"0"
train_log$class[train_log$class=="pos"]<-"1"
train_log$class<-as.factor(train_log$class)
levels(train_log$class)<- c('neg', 'pos')

summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
prop.table(table(train_log$class))
```

```
##
##        neg        pos
## 0.98330148 0.01669852
```

```
table(train_log$class)
```

```
##
##   neg   pos
## 58650   996
```

```
library(caret)

model_log1 <- glm (class~., data=train_log, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_log1)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial, data = train_log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3695  -0.0401  -0.0155  -0.0004   6.6093
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.430e+01  1.984e+00 -17.284  < 2e-16 ***
## aa_000       3.142e-01  1.108e-01   2.836 0.004572 **
## ac_000      -1.741e-02  2.297e-02  -0.758 0.448326
## ag_000       4.342e-02  5.839e-02   0.744 0.457088
## ag_002       1.527e-02  3.107e-02   0.491 0.623164
## ag_004      -2.175e-01  9.109e-02  -2.387 0.016976 *
## ag_005      -5.531e-02  1.510e-01  -0.366 0.714158
## ag_006      -1.601e-01  1.576e-01  -1.016 0.309659
## ag_007      -9.659e-02  7.809e-02  -1.237 0.216126
## ag_008      -9.976e-03  3.983e-02  -0.251 0.802200
## ag_009      -5.414e-02  2.120e-02  -2.553 0.010665 *
## aj_000       9.874e-03  2.012e-02   0.491 0.623601
## al_000      -1.023e-01  7.190e-02  -1.423 0.154803
## am_0         9.869e-02  6.582e-02   1.499 0.133795
## an_000       8.538e-01  5.715e-01   1.494 0.135225
## ao_000       6.774e-01  3.077e-01   2.201 0.027713 *
## ap_000       2.704e+00  4.367e-01   6.192 5.95e-10 ***
## aq_000       1.729e+00  3.111e-01   5.558 2.73e-08 ***
## ar_000      -9.365e-02  8.483e-02  -1.104 0.269633
## as_000       2.049e-02  7.151e-02   0.287 0.774435
## at_000      -4.133e-02  1.935e-02  -2.136 0.032701 *
## au_000       6.422e-02  6.218e-02   1.033 0.301632
## av_000       4.863e-01  9.417e-02   5.163 2.42e-07 ***
## ax_000      -5.077e-01  1.084e-01  -4.684 2.81e-06 ***
## ay_000       2.414e-01  5.665e-02   4.262 2.03e-05 ***
## ay_001      -2.563e-01  9.729e-02  -2.635 0.008417 **
## ay_002       3.329e-01  9.224e-02   3.609 0.000308 ***
## ay_003      -1.442e-01  5.498e-02  -2.622 0.008740 **
## ay_004       7.587e-02  2.435e-02   3.115 0.001839 **
## ay_005       1.448e-02  1.935e-02   0.748 0.454369
## ay_006      -1.353e-01  2.173e-02  -6.226 4.78e-10 ***
## ay_007      -1.198e-02  4.836e-02  -0.248 0.804373
## ay_008       8.500e-02  3.315e-02   2.564 0.010353 *
## ay_009       1.412e-01  3.037e-02   4.650 3.32e-06 ***
## az_000       2.016e-01  1.400e-01   1.440 0.149918
## az_001       2.328e-01  2.037e-01   1.143 0.252993
## az_002       4.818e-02  1.820e-01   0.265 0.791189
## az_003       9.092e-02  8.202e-02   1.109 0.267637
## az_004      -9.982e-02  5.929e-02  -1.684 0.092266 .
## az_005       1.070e-01  1.050e-01   1.019 0.308318
## az_006      -4.114e-02  2.499e-02  -1.646 0.099688 .
## az_007       2.439e-03  3.185e-02   0.077 0.938958
## az_008      -2.850e-02  3.900e-02  -0.731 0.464884
## az_009       4.784e-02  6.331e-02   0.756 0.449854
## ba_002      -6.922e-01  3.248e-01  -2.131 0.033101 *
## ba_003       3.839e-01  3.145e-01   1.221 0.222229
## ba_008      -6.465e-02  2.740e-02  -2.359 0.018313 *
## ba_009       4.018e-02  2.616e-02   1.536 0.124502
## bb_000      -7.355e-01  4.352e-01  -1.690 0.091048 .
## bc_000       8.083e-02  4.056e-02   1.993 0.046262 *
## bd_000       1.834e-01  5.487e-02   3.343 0.000828 ***
## be_000      -1.365e-01  6.457e-02  -2.115 0.034464 *
## bf_000      -6.810e-03  3.294e-02  -0.207 0.836225
## bg_000      -1.014e+00  3.470e-01  -2.922 0.003476 **
## bh_000      -2.500e+00  5.296e-01  -4.721 2.34e-06 ***
## bi_000      -1.783e+00  2.483e-01  -7.184 6.79e-13 ***
## bj_000      -6.336e-01  3.175e-01  -1.995 0.046003 *
## bu_000       1.240e+06  3.437e+05   3.607 0.000310 ***
## bv_000       7.697e+03  4.128e+03   1.865 0.062241 .
## bx_000       4.255e-01  1.478e-01   2.879 0.003994 **
## ca_000       2.690e-02  7.281e-02   0.370 0.711733
```

```
## cb_000      -2.234e-02  5.529e-02  -0.404 0.686175
## cc_000      -1.796e-01  1.031e-01  -1.742 0.081533 .
## ce_000      -2.132e-02  2.175e-02  -0.980 0.326997
## ci_000       1.635e+00  2.242e-01   7.293 3.04e-13 ***
## cj_000      -1.312e-02  1.537e-02  -0.854 0.393298
## ck_000      -2.265e-01  1.837e-01  -1.233 0.217457
## cl_000       1.069e-01  3.505e-02   3.050 0.002290 **
## cm_000      -1.417e-01  4.124e-02  -3.436 0.000591 ***
## cn_000       1.517e-01  2.923e-02   5.189 2.12e-07 ***
## cn_001      -2.130e-02  2.850e-02  -0.748 0.454750
## cn_002       1.605e-01  4.849e-02   3.310 0.000933 ***
## cn_003       7.442e-02  1.311e-01   0.568 0.570205
## cn_004       7.258e-01  2.838e-01   2.557 0.010550 *
## cn_005       2.777e-01  2.614e-01   1.062 0.288127
## cn_006      -2.210e-01  2.277e-01  -0.971 0.331750
## cn_007       2.003e-02  2.307e-01   0.087 0.930786
## cn_008       5.848e-02  1.583e-01   0.370 0.711734
## cn_009       1.108e-01  5.300e-02   2.090 0.036583 *
## cp_000      -4.929e-02  2.979e-02  -1.655 0.098002 .
## cq_000      -1.247e+06  3.438e+05  -3.629 0.000285 ***
## cs_000      -9.018e-01  2.848e-01  -3.167 0.001540 **
## cs_001       1.221e-02  2.338e-01   0.052 0.958335
## cs_002      -1.036e-01  6.779e-02  -1.528 0.126478
## cs_003       2.327e-01  1.540e-01   1.511 0.130798
## cs_004       2.448e-01  1.220e-01   2.006 0.044821 *
## cs_005      -2.105e-01  1.460e-01  -1.442 0.149384
## cs_006      -3.362e-02  8.466e-02  -0.397 0.691276
## cs_007      -2.385e-02  4.385e-02  -0.544 0.586587
## cs_008      -3.428e-02  4.289e-02  -0.799 0.424085
## cs_009       1.068e-01  9.974e-02   1.071 0.284055
## dd_000      -3.047e-02  8.264e-02  -0.369 0.712378
## de_000      -9.003e-02  7.555e-02  -1.192 0.233408
## df_000      -3.512e-01  1.943e-01  -1.807 0.070726 .
## dg_000       4.286e-01  1.728e-01   2.480 0.013139 *
## dh_000      -1.407e-01  5.596e-02  -2.515 0.011898 *
## di_000       1.029e-01  3.653e-02   2.816 0.004862 **
## dj_000       4.939e-02  3.979e-01   0.124 0.901212
## dk_000       1.665e-02  1.615e-01   0.103 0.917875
## dl_000      -7.381e-01  6.745e-01  -1.094 0.273817
## dm_000       5.878e-01  5.425e-01   1.083 0.278615
## dn_000       1.478e+00  4.233e-01   3.492 0.000480 ***
## do_000      -2.046e-01  2.843e-01  -0.720 0.471663
## dp_000       2.229e-01  2.846e-01   0.783 0.433502
## dq_000       3.555e-02  3.302e-02   1.076 0.281770
## dr_000      -9.738e-02  4.287e-02  -2.272 0.023114 *
## ds_000      -1.370e-01  2.473e-01  -0.554 0.579673
## dt_000       5.087e-01  2.745e-01   1.853 0.063875 .
## du_000      -3.212e-02  1.075e-01  -0.299 0.765190
## dv_000      -1.514e-01  1.153e-01  -1.313 0.189283
## dx_000      -2.685e-02  2.534e-02  -1.059 0.289431
## dy_000       2.013e-02  3.336e-02   0.603 0.546281
## dz_000       4.799e-01  3.170e-01   1.514 0.130082
## ea_000      -2.409e-01  2.292e-01  -1.051 0.293317
## eb_000      -5.853e-03  1.724e-02  -0.339 0.734260
## ec_00       -3.917e-01  8.428e-02  -4.647 3.37e-06 ***
## ed_000       4.785e-01  9.277e-02   5.158 2.49e-07 ***
## ee_000      -1.395e-01  2.255e-01  -0.618 0.536341
## ee_001      -1.360e-01  2.352e-01  -0.578 0.563218
## ee_002       1.779e-01  1.984e-01   0.897 0.369884
## ee_003      -4.471e-02  2.382e-01  -0.188 0.851125
## ee_004      -5.757e-01  2.275e-01  -2.531 0.011380 *
## ee_005       5.015e-01  1.294e-01   3.877 0.000106 ***
## ee_006       1.825e-01  8.671e-02   2.105 0.035292 *
## ee_007      -5.011e-01  7.105e-02  -7.052 1.76e-12 ***
## ee_008       2.822e-01  6.106e-02   4.621 3.81e-06 ***
## ee_009      -3.773e-02  2.779e-02  -1.358 0.174583
## ef_000       8.942e-02  4.245e-01   0.211 0.833185
## eg_000      -1.143e-01  3.361e-01  -0.340 0.733831
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10127.4  on 59645  degrees of freedom
## Residual deviance:  2642.9  on 59517  degrees of freedom
## AIC: 2900.9
##
## Number of Fisher Scoring iterations: 13
```

```
# Mostrar las variables significativas (p<0.05)
sig_coeff<-data.frame(summary(model_log1)$coef[summary(model_log1)$coef[,4] <= .05,c(4)])
names(sig_coeff)<-c("p_value")
dim(sig_coeff)[1]
```

```
## [1] 51
```

```
rownames(sig_coeff)
```

```
##  [1] "(Intercept)" "aa_000"      "ag_004"      "ag_009"      "ao_000"
##  [6] "ap_000"      "aq_000"      "at_000"      "av_000"      "ax_000"
## [11] "ay_000"      "ay_001"      "ay_002"      "ay_003"      "ay_004"
## [16] "ay_006"      "ay_008"      "ay_009"      "ba_002"      "ba_008"
## [21] "bc_000"      "bd_000"      "be_000"      "bg_000"      "bh_000"
## [26] "bi_000"      "bj_000"      "bu_000"      "bx_000"      "ci_000"
## [31] "cl_000"      "cm_000"      "cn_000"      "cn_002"      "cn_004"
## [36] "cn_009"      "cq_000"      "cs_000"      "cs_004"      "dg_000"
## [41] "dh_000"      "di_000"      "dn_000"      "dr_000"      "ec_00"
## [46] "ed_000"      "ee_004"      "ee_005"      "ee_006"      "ee_007"
## [51] "ee_008"
```

```
## Predicción de valores
predict <- predict(model_log1, test_log, type = 'response')

## Matriz de confusión
cm<-table(test_log$class, predict > 0.5)
cm
```

```
##
##       FALSE  TRUE
##   neg 15581    44
##   pos   127   248
```

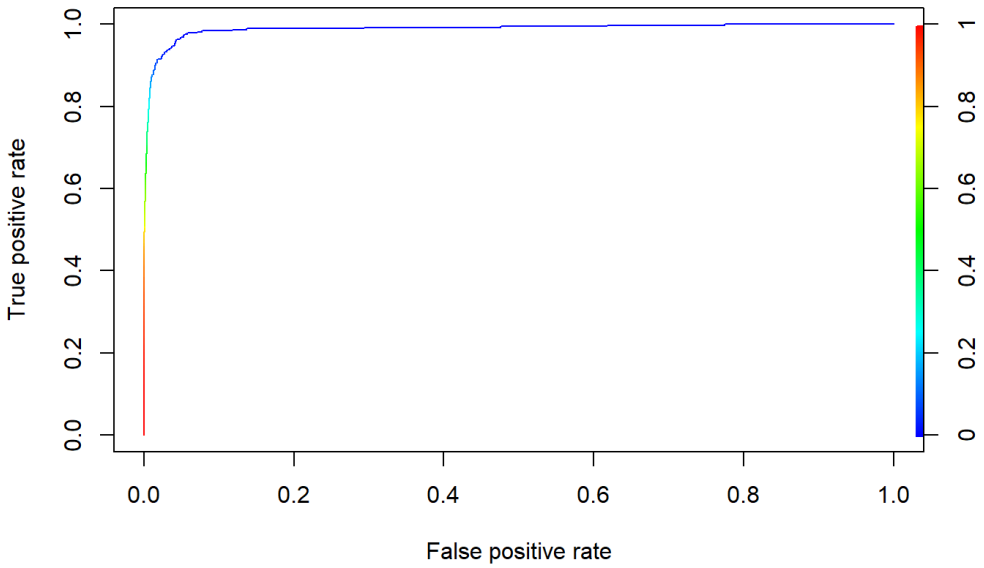
```
precision<-(cm[1,1]+cm[2,2])/sum(cm);precision
```

```
## [1] 0.9893125
```

```
## Coste Total
coste_total_log <- cm[1,2]*10 + cm[2,1]*500
coste_total_log
```

```
## [1] 63940
```

```
#ROCR Curve
library(ROCR)
ROCRpred <- prediction(predict, test_log$class)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```




```
# AUC value
auc_ROCR <- performance(ROCRpred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]];auc_ROCR
```

```
## [1] 0.9885594
```

5.4.2.3 Regresión logística, imputación mediana, datos transformados Box-Cox y con datos balanceados mediante SMOTE

Se continúa este caso con datos transformados Box-Cot con imputación con la mediana. Balanceado de datos con SMOTE. Se utilizan como parámetros perc.over=300 y perc.under=150. El perc.over amplifica el número de clases de la clase minoritaria, mientras que perc.under aminora la mayoritaria. De esta forma, se balancea la proporción y se obtienen 4482 negativos y 3984 positivos.

Se obtiene un modelo con 55 variables significativas. La precisión del modelo es 96.12%. Hay 22 falsos negativos. Solo el 5.8% de los casos positivos están mal clasificados, una mejora muy significativa. El coste total basado en la matriz de confusión es 16980. El indicador AIC (cuanto menor mejor) es 1839.

```
library(DMwR)
train_log<-train_bc_median
test_log<-test_bc_median
train_log$class[train_log$class=="neg"]<-"0"
train_log$class[train_log$class=="pos"]<-"1"
train_log$class<-as.factor(train_log$class)
levels(train_log$class)<- c('neg', 'pos')

summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
prop.table(table(train_log$class))
```

```
##
##        neg        pos
## 0.98330148 0.01669852
```

```
## Smote : Synthetic Minority Oversampling Technique To Handle Class Imbalancy In Binary Classification
train_balanced <- SMOTE(class ~., train_log, perc.over = 300, k = 5, perc.under = 150)
summary(train_balanced$class)
```

```
##  neg  pos
## 4482 3984
```

```
library(caret)
ctrl<-glm.control(epsilon = 1e-8, maxit =100, trace = FALSE)
model_log2 <- glm (class~., data=train_balanced, family = binomial,control=ctrl)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_log2)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial, data = train_balanced,
##     control = ctrl)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.4243  -0.0642  -0.0002   0.0657   4.3928
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.798e+01  2.382e+00 -11.747  < 2e-16 ***
## aa_000       3.765e-01  1.477e-01   2.549 0.010800 *
## ac_000       9.116e-02  2.314e-02   3.939 8.17e-05 ***
## ag_000       3.060e-01  1.680e-01   1.822 0.068495 .
## ag_002      -1.766e-01  5.608e-02  -3.148 0.001642 **
## ag_004      -1.376e-01  1.043e-01  -1.319 0.187142
## ag_005      -9.229e-01  1.583e-01  -5.829 5.58e-09 ***
## ag_006       2.504e-01  1.958e-01   1.279 0.200947
## ag_007      -2.758e-01  1.035e-01  -2.666 0.007675 **
## ag_008       5.296e-02  5.361e-02   0.988 0.323194
## ag_009      -1.443e-01  3.247e-02  -4.445 8.78e-06 ***
## aj_000       6.700e-02  3.219e-02   2.082 0.037376 *
## al_000       1.664e-01  1.536e-01   1.083 0.278645
## am_0        -2.028e-01  1.439e-01  -1.409 0.158859
## an_000      -2.417e+00  9.460e-01  -2.555 0.010631 *
## ao_000       1.195e+00  6.320e-01   1.891 0.058665 .
## ap_000       3.959e+00  7.075e-01   5.595 2.21e-08 ***
## aq_000       1.214e+00  4.108e-01   2.954 0.003134 **
## ar_000      -3.573e-01  1.701e-01  -2.101 0.035660 *
## as_000      -4.003e-03  1.862e-01  -0.021 0.982850
## at_000       5.257e-02  3.413e-02   1.540 0.123563
## au_000       3.345e-03  1.429e-01   0.023 0.981323
## av_000       4.272e-01  1.147e-01   3.724 0.000196 ***
## ax_000      -7.322e-01  1.307e-01  -5.602 2.12e-08 ***
## ay_000       5.981e-01  1.086e-01   5.508 3.63e-08 ***
## ay_001      -6.755e-01  2.012e-01  -3.357 0.000788 ***
## ay_002       6.094e-01  1.906e-01   3.198 0.001383 **
## ay_003      -2.031e-01  1.007e-01  -2.018 0.043602 *
## ay_004       4.694e-02  3.946e-02   1.190 0.234151
## ay_005      -1.850e-02  2.792e-02  -0.662 0.507742
## ay_006      -2.655e-02  3.433e-02  -0.773 0.439274
## ay_007      -7.893e-02  5.917e-02  -1.334 0.182218
## ay_008       1.357e-01  3.957e-02   3.429 0.000606 ***
## ay_009       2.866e-01  7.955e-02   3.603 0.000315 ***
## az_000       8.074e-01  2.289e-01   3.528 0.000418 ***
## az_001      -5.670e-01  2.876e-01  -1.972 0.048658 *
## az_002       5.645e-01  2.688e-01   2.100 0.035759 *
## az_003       1.850e-01  1.069e-01   1.731 0.083480 .
## az_004      -1.690e-01  8.606e-02  -1.964 0.049498 *
## az_005       2.477e-01  1.374e-01   1.804 0.071294 .
## az_006      -6.437e-02  3.463e-02  -1.859 0.063075 .
## az_007      -8.751e-02  4.396e-02  -1.991 0.046524 *
## az_008      -1.756e-02  6.440e-02  -0.273 0.785135
## az_009       8.337e-02  1.194e-01   0.698 0.484941
## ba_002      -2.851e-01  4.584e-01  -0.622 0.534012
## ba_003      -3.462e-02  4.354e-01  -0.080 0.936626
## ba_008      -1.578e-01  3.569e-02  -4.422 9.76e-06 ***
## ba_009       1.279e-01  3.832e-02   3.337 0.000848 ***
## bb_000       1.235e-01  5.363e-01   0.230 0.817849
## bc_000       2.744e-01  4.851e-02   5.656 1.55e-08 ***
## bd_000       2.034e-02  6.533e-02   0.311 0.755518
## be_000      -4.189e-01  8.427e-02  -4.970 6.68e-07 ***
## bf_000      -1.971e-02  5.244e-02  -0.376 0.706986
## bg_000       1.119e+00  5.952e-01   1.881 0.060020 .
## bh_000      -3.512e+00  7.413e-01  -4.737 2.16e-06 ***
## bi_000      -1.796e+00  3.674e-01  -4.888 1.02e-06 ***
## bj_000      -3.948e-01  4.348e-01  -0.908 0.363883
## bu_000       2.592e+04  3.676e+05   0.071 0.943795
## bv_000       7.298e+03  2.594e+04   0.281 0.778452
## bx_000       6.874e-01  1.873e-01   3.670 0.000242 ***
```

```
## ca_000      -1.103e-01  8.802e-02  -1.253 0.210289
## cb_000      -5.228e-02  8.447e-02  -0.619 0.535998
## cc_000      -2.843e-01  1.339e-01  -2.124 0.033687 *
## ce_000      -5.906e-02  3.213e-02  -1.838 0.066069 .
## ci_000       1.949e+00  3.438e-01   5.667 1.45e-08 ***
## cj_000      -2.482e-02  2.746e-02  -0.904 0.366193
## ck_000      -9.912e-01  3.002e-01  -3.302 0.000960 ***
## cl_000       8.407e-02  5.922e-02   1.420 0.155740
## cm_000      -8.115e-02  6.333e-02  -1.281 0.200082
## cn_000       4.257e-01  5.958e-02   7.145 9.01e-13 ***
## cn_001      -2.325e-02  4.374e-02  -0.532 0.594999
## cn_002       2.444e-01  5.187e-02   4.712 2.46e-06 ***
## cn_003       4.826e-01  1.599e-01   3.019 0.002540 **
## cn_004       1.774e+00  3.577e-01   4.960 7.06e-07 ***
## cn_005       1.300e-01  3.218e-01   0.404 0.686223
## cn_006       3.566e-01  2.737e-01   1.303 0.192627
## cn_007       2.800e-01  2.392e-01   1.170 0.241885
## cn_008      -3.651e-01  1.709e-01  -2.137 0.032624 *
## cn_009       2.702e-01  6.797e-02   3.975 7.03e-05 ***
## cp_000       3.664e-02  4.496e-02   0.815 0.415154
## cq_000      -3.322e+04  3.686e+05  -0.090 0.928192
## cs_000      -1.363e+00  3.642e-01  -3.742 0.000182 ***
## cs_001       4.838e-01  3.095e-01   1.563 0.117969
## cs_002      -2.098e-01  8.353e-02  -2.512 0.012019 *
## cs_003       2.832e-01  2.035e-01   1.391 0.164173
## cs_004       2.521e-01  1.852e-01   1.362 0.173304
## cs_005      -5.134e-01  1.943e-01  -2.642 0.008246 **
## cs_006      -1.702e-01  1.369e-01  -1.244 0.213595
## cs_007       1.192e-02  7.633e-02   0.156 0.875955
## cs_008      -4.253e-02  5.853e-02  -0.727 0.467509
## cs_009      -2.103e-01  1.480e-01  -1.421 0.155363
## dd_000       3.259e-01  1.192e-01   2.734 0.006258 **
## de_000       1.376e-01  1.057e-01   1.302 0.192888
## df_000       1.345e-01  2.569e-01   0.523 0.600649
## dg_000       1.543e-01  1.825e-01   0.846 0.397797
## dh_000       2.149e-03  8.096e-02   0.027 0.978824
## di_000       5.597e-02  5.358e-02   1.045 0.296183
## dj_000      -7.669e-01  1.123e+00  -0.683 0.494618
## dk_000       1.013e-01  4.988e-01   0.203 0.839062
## dl_000      -4.355e-01  8.190e-01  -0.532 0.594902
## dm_000       3.013e-01  7.374e-01   0.409 0.682785
## dn_000       2.140e+00  5.697e-01   3.756 0.000173 ***
## do_000      -6.245e-01  2.781e-01  -2.245 0.024757 *
## dp_000       3.283e-01  3.151e-01   1.042 0.297350
## dq_000      -1.584e-01  1.093e-01  -1.450 0.147157
## dr_000       9.429e-02  1.265e-01   0.745 0.456131
## ds_000       2.613e-01  2.822e-01   0.926 0.354384
## dt_000       4.165e-01  3.336e-01   1.248 0.211899
## du_000      -1.416e-02  1.645e-01  -0.086 0.931424
## dv_000      -2.873e-01  1.808e-01  -1.590 0.111928
## dx_000       3.828e-02  3.325e-02   1.151 0.249591
## dy_000      -6.028e-02  4.917e-02  -1.226 0.220250
## dz_000       4.826e-01  4.514e-01   1.069 0.284962
## ea_000      -4.539e-01  3.377e-01  -1.344 0.178890
## eb_000       1.400e-02  1.996e-02   0.701 0.483050
## ec_00       -3.113e-01  1.050e-01  -2.964 0.003038 **
## ed_000       4.199e-01  1.230e-01   3.413 0.000643 ***
## ee_000      -5.075e-01  3.523e-01  -1.440 0.149776
## ee_001      -4.663e-01  3.415e-01  -1.366 0.172091
## ee_002      -9.013e-01  3.663e-01  -2.460 0.013883 *
## ee_003       9.401e-01  3.504e-01   2.683 0.007302 **
## ee_004      -7.846e-01  2.915e-01  -2.691 0.007123 **
## ee_005      -1.466e-01  1.877e-01  -0.781 0.434786
## ee_006       5.682e-01  1.353e-01   4.199 2.69e-05 ***
## ee_007      -5.127e-01  1.048e-01  -4.892 9.96e-07 ***
## ee_008       2.444e-01  8.713e-02   2.805 0.005027 **
## ee_009      -5.705e-02  3.852e-02  -1.481 0.138661
## ef_000      -1.062e-01  6.614e-01  -0.161 0.872379
## eg_000      -1.805e-01  5.740e-01  -0.314 0.753145
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11707  on 8465  degrees of freedom
## Residual deviance:  1581  on 8337  degrees of freedom
## AIC: 1839
##
## Number of Fisher Scoring iterations: 9
```

```
# Mostrar las variables significativas (p<0.05)
sig_coeff2<-data.frame(summary(model_log2)$coef[summary(model_log2)$coef[,4] <= .05,c(4)])
names(sig_coeff2)<-c("p_value")
dim(sig_coeff2)[1]
```

```
## [1] 55
```

```
rownames(sig_coeff2)
```

```
##  [1] "(Intercept)" "aa_000"      "ac_000"      "ag_002"      "ag_005"
##  [6] "ag_007"      "ag_009"      "aj_000"      "an_000"      "ap_000"
## [11] "aq_000"      "ar_000"      "av_000"      "ax_000"      "ay_000"
## [16] "ay_001"      "ay_002"      "ay_003"      "ay_008"      "ay_009"
## [21] "az_000"      "az_001"      "az_002"      "az_004"      "az_007"
## [26] "ba_008"      "ba_009"      "bc_000"      "be_000"      "bh_000"
## [31] "bi_000"      "bx_000"      "cc_000"      "ci_000"      "ck_000"
## [36] "cn_000"      "cn_002"      "cn_003"      "cn_004"      "cn_008"
## [41] "cn_009"      "cs_000"      "cs_002"      "cs_005"      "dd_000"
## [46] "dn_000"      "do_000"      "ec_00"       "ed_000"      "ee_002"
## [51] "ee_003"      "ee_004"      "ee_006"      "ee_007"      "ee_008"
```

```
## Predicción de valores
predict <- predict(model_log2, test_log, type = 'response')

## Matriz de confusión
cm<-table(test_log$class, predict > 0.5)
cm
```

```
##
##       FALSE  TRUE
##   neg 15027   598
##   pos    22   353
```

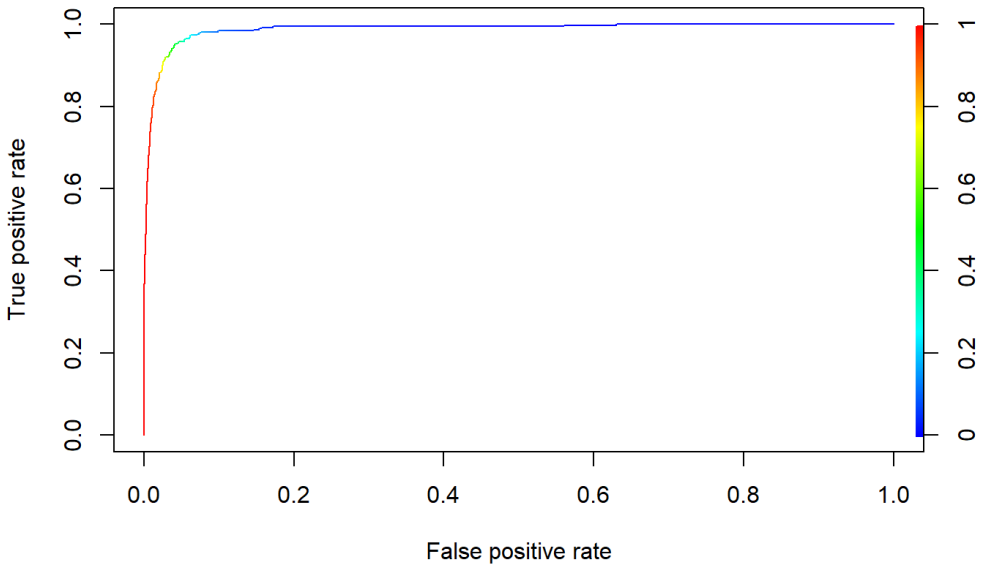
```
precision<-(cm[1,1]+cm[2,2])/sum(cm);precision
```

```
## [1] 0.96125
```

```
## Coste Total
coste_total_log <- cm[1,2]*10 + cm[2,1]*500
coste_total_log
```

```
## [1] 16980
```

```
#ROCR Curve
library(ROCR)
ROCRpred <- prediction(predict, test_log$class)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))
```




```
# AUC value
auc_ROCR <- performance(ROCRpred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]];auc_ROCR
```

```
## [1] 0.9875581
```

5.4.3 Regresión logística, imputación mediana, datos transformados Box-Cox y con datos balanceados mediante pesos

Se continúa este caso con datos transformados Box-Cox con imputación por mediana. Balanceado de datos con Weights, de forma que el peso de todos los negativos es 0.5 y todos los positivos es 0.5 (suma total de pesos igual a 1). El hecho de incrementar el peso de los positivos hace que el modelo resultante sea más sensible al error de falso negativo. En este caso ninguna de las variables predictoras resulta ser significativa, si bien el modelo se ajusta bien a los datos de testeo y el coste total es muy bajo. Sin embargo, que ninguna variable predictora resulte ser significativa es un mal resultado. EL número de iteraciones para llegar al resultado ha sido 12 y el AIC final de 258. El coste total es 16650.

Si en vez de utilizar los datos transformados Box-Cox se utilizan los datos sin transformar, al modelo le cuesta converger (número de iteraciones máximo impuesto 100) y resulta que las 129 variables son significativas, hecho que tampoco es deseble.

Se muestra la ejecución con datos transformados Box-Cox.

```
train_log<-train_bc_median
test_log<-test_bc_median
train_log$class[train_log$class=="neg"]<-"0"
train_log$class[train_log$class=="pos"]<-"1"
train_log$class<-as.factor(train_log$class)
levels(train_log$class)<- c('neg', 'pos')

summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
# Create model weights (they sum to one)

model_weights <- ifelse(train_log$class == "neg",
                        (1/table(train_log$class)[1]) * 0.5,
                        (1/table(train_log$class)[2]) * 0.5)
summary(train_log$class)
```

```
##   neg   pos
## 58650   996
```

```
sum(model_weights)
```

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

```
library(caret)
ctrl<-glm.control(epsilon = 1e-8, maxit =100, trace = FALSE)
model_log3 <- glm (class~.,data=train_log, weights=model_weights,family = binomial,control=ctrl)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_log3)
```

```
##
## Call:
## glm(formula = class ~ ., family = binomial, data = train_log,
##     weights = model_weights, control = ctrl)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.017096  -0.000702  -0.000283  -0.000055   0.084759
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.434e+01  1.916e+02  -0.127    0.899
## aa_000       2.088e-01  9.395e+00   0.022    0.982
## ac_000       4.183e-02  1.780e+00   0.023    0.981
## ag_000       1.205e-01  9.128e+00   0.013    0.989
## ag_002      -8.790e-02  3.972e+00  -0.022    0.982
## ag_004      -4.854e-02  8.407e+00  -0.006    0.995
## ag_005      -5.510e-01  1.223e+01  -0.045    0.964
## ag_006       8.947e-02  1.403e+01   0.006    0.995
## ag_007      -1.029e-01  7.443e+00  -0.014    0.989
## ag_008       4.468e-02  4.014e+00   0.011    0.991
## ag_009      -1.137e-01  2.447e+00  -0.046    0.963
## aj_000       6.444e-02  2.465e+00   0.026    0.979
## al_000      -7.827e-02  1.091e+01  -0.007    0.994
## am_0         1.011e-02  1.023e+01   0.001    0.999
## an_000      -1.382e+00  6.443e+01  -0.021    0.983
## ao_000       9.539e-01  5.016e+01   0.019    0.985
## ap_000       3.607e+00  5.833e+01   0.062    0.951
## aq_000       1.337e+00  3.070e+01   0.044    0.965
## ar_000      -3.228e-01  1.162e+01  -0.028    0.978
## as_000      -4.818e-02  1.635e+01  -0.003    0.998
## at_000       1.202e-02  2.404e+00   0.005    0.996
## au_000       1.590e-01  1.383e+01   0.011    0.991
## av_000       4.298e-01  9.024e+00   0.048    0.962
## ax_000      -6.101e-01  1.012e+01  -0.060    0.952
## ay_000       3.794e-01  7.641e+00   0.050    0.960
## ay_001      -5.028e-01  1.411e+01  -0.036    0.972
## ay_002       5.787e-01  1.304e+01   0.044    0.965
## ay_003      -1.846e-01  7.170e+00  -0.026    0.979
## ay_004       4.261e-02  3.168e+00   0.013    0.989
## ay_005       2.593e-03  2.096e+00   0.001    0.999
## ay_006      -6.221e-02  2.605e+00  -0.024    0.981
## ay_007      -9.529e-02  4.630e+00  -0.021    0.984
## ay_008       5.955e-02  2.869e+00   0.021    0.983
## ay_009       1.823e-01  5.196e+00   0.035    0.972
## az_000       5.238e-01  1.706e+01   0.031    0.976
## az_001      -2.237e-01  2.146e+01  -0.010    0.992
## az_002       3.572e-01  2.007e+01   0.018    0.986
## az_003       1.011e-01  8.246e+00   0.012    0.990
## az_004      -2.109e-01  6.414e+00  -0.033    0.974
## az_005       1.870e-01  1.133e+01   0.017    0.987
## az_006      -4.730e-02  2.613e+00  -0.018    0.986
## az_007      -4.430e-02  3.422e+00  -0.013    0.990
## az_008      -3.054e-02  4.880e+00  -0.006    0.995
## az_009      -4.930e-02  8.982e+00  -0.005    0.996
## ba_002      -4.082e-01  3.383e+01  -0.012    0.990
## ba_003       4.996e-02  3.150e+01   0.002    0.999
## ba_008      -9.085e-02  2.712e+00  -0.033    0.973
## ba_009       9.444e-02  2.840e+00   0.033    0.973
## bb_000      -3.950e-01  4.649e+01  -0.008    0.993
## bc_000       2.148e-01  3.792e+00   0.057    0.955
## bd_000       4.936e-02  4.985e+00   0.010    0.992
## be_000      -2.429e-01  6.310e+00  -0.038    0.969
## bf_000      -6.038e-02  3.900e+00  -0.015    0.988
## bg_000      -2.080e-01  4.031e+01  -0.005    0.996
## bh_000      -2.019e+00  5.306e+01  -0.038    0.970
## bi_000      -1.978e+00  3.196e+01  -0.062    0.951
## bj_000      -9.003e-01  3.816e+01  -0.024    0.981
## bu_000       1.727e+05  4.703e+07   0.004    0.997
## bv_000       3.720e+03  4.259e+05   0.009    0.993
## bx_000       7.927e-01  1.800e+01   0.044    0.965
```

```
## ca_000      -4.166e-02  7.010e+00  -0.006    0.995
## cb_000      -1.957e-01  6.218e+00  -0.031    0.975
## cc_000      -3.497e-01  1.346e+01  -0.026    0.979
## ce_000      -4.244e-02  2.467e+00  -0.017    0.986
## ci_000       1.879e+00  2.629e+01   0.071    0.943
## cj_000      -4.462e-02  2.178e+00  -0.020    0.984
## ck_000      -8.612e-01  2.287e+01  -0.038    0.970
## cl_000       1.323e-01  4.653e+00   0.028    0.977
## cm_000      -1.401e-01  4.642e+00  -0.030    0.976
## cn_000       2.969e-01  4.153e+00   0.072    0.943
## cn_001      -2.980e-02  3.270e+00  -0.009    0.993
## cn_002       2.069e-01  4.032e+00   0.051    0.959
## cn_003       1.834e-01  1.168e+01   0.016    0.987
## cn_004       1.386e+00  2.606e+01   0.053    0.958
## cn_005      -3.642e-02  2.510e+01  -0.001    0.999
## cn_006       1.937e-01  2.000e+01   0.010    0.992
## cn_007       3.708e-01  1.845e+01   0.020    0.984
## cn_008      -6.890e-01  1.310e+01  -0.053    0.958
## cn_009       3.314e-01  5.404e+00   0.061    0.951
## cp_000      -2.864e-04  3.311e+00   0.000    1.000
## cq_000      -1.764e+05  4.703e+07  -0.004    0.997
## cs_000      -7.446e-01  2.678e+01  -0.028    0.978
## cs_001       6.174e-02  2.349e+01   0.003    0.998
## cs_002      -1.040e-01  6.828e+00  -0.015    0.988
## cs_003      -5.981e-02  1.631e+01  -0.004    0.997
## cs_004       2.889e-02  1.499e+01   0.002    0.998
## cs_005      -7.143e-01  1.639e+01  -0.044    0.965
## cs_006       2.309e-02  1.024e+01   0.002    0.998
## cs_007      -3.233e-02  5.864e+00  -0.006    0.996
## cs_008      -1.341e-01  4.601e+00  -0.029    0.977
## cs_009       1.587e-01  1.383e+01   0.011    0.991
## dd_000       1.762e-01  9.195e+00   0.019    0.985
## de_000       8.150e-02  8.195e+00   0.010    0.992
## df_000       1.210e-01  1.775e+01   0.007    0.995
## dg_000       3.979e-02  1.390e+01   0.003    0.998
## dh_000      -2.755e-02  6.493e+00  -0.004    0.997
## di_000       4.252e-02  4.284e+00   0.010    0.992
## dj_000      -4.983e-02  5.635e+01  -0.001    0.999
## dk_000      -7.986e-02  2.251e+01  -0.004    0.997
## dl_000      -8.517e-01  7.073e+01  -0.012    0.990
## dm_000       4.694e-01  5.736e+01   0.008    0.993
## dn_000       1.775e+00  4.100e+01   0.043    0.965
## do_000      -7.206e-01  2.339e+01  -0.031    0.975
## dp_000       5.456e-01  2.475e+01   0.022    0.982
## dq_000      -3.202e-02  5.067e+00  -0.006    0.995
## dr_000      -1.983e-02  5.940e+00  -0.003    0.997
## ds_000       2.220e-01  2.415e+01   0.009    0.993
## dt_000       2.771e-01  2.775e+01   0.010    0.992
## du_000       4.634e-02  1.203e+01   0.004    0.997
## dv_000      -2.693e-01  1.306e+01  -0.021    0.984
## dx_000       1.426e-02  2.511e+00   0.006    0.995
## dy_000      -6.107e-02  3.688e+00  -0.017    0.987
## dz_000       6.749e-01  3.499e+01   0.019    0.985
## ea_000      -6.556e-01  2.425e+01  -0.027    0.978
## eb_000      -1.647e-02  1.558e+00  -0.011    0.992
## ec_00       -3.108e-01  8.122e+00  -0.038    0.969
## ed_000       3.889e-01  9.442e+00   0.041    0.967
## ee_000       2.228e-01  2.708e+01   0.008    0.993
## ee_001      -1.156e-01  2.567e+01  -0.005    0.996
## ee_002      -4.082e-01  2.824e+01  -0.014    0.988
## ee_003       3.424e-01  2.851e+01   0.012    0.990
## ee_004      -5.780e-01  2.338e+01  -0.025    0.980
## ee_005       1.401e-01  1.490e+01   0.009    0.992
## ee_006       4.285e-01  9.790e+00   0.044    0.965
## ee_007      -3.625e-01  7.938e+00  -0.046    0.964
## ee_008       1.713e-01  6.746e+00   0.025    0.980
## ee_009      -4.409e-02  2.928e+00  -0.015    0.988
## ef_000       1.933e-01  5.006e+01   0.004    0.997
## eg_000       1.742e-01  3.998e+01   0.004    0.997
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 1.38629  on 59645  degrees of freedom
## Residual deviance: 0.22965  on 59517  degrees of freedom
## AIC: 258
##
## Number of Fisher Scoring iterations: 12
```

En la matriz de confusión, los falsos negativos se reducen a 23, aunque hayan aumentado los falsos positivos el coste total se reduce a 17770.

```
# Mostrar las variables significativas (p<0.05)
sig_coeff3<-data.frame(summary(model_log3)$coef[summary(model_log3)$coef[,4] <= .05,c(4)])
names(sig_coeff3)<-c("p_value")
dim(sig_coeff3)[1]
```

```
## [1] 0
```

```
rownames(sig_coeff3)
```

```
## character(0)
```

```
## Predicción de valores
predict <- predict(model_log3, test_log, type = 'response')

## Matriz de confusión
cm<-table(test_log$class, predict > 0.5)
cm
```

```
##
##       FALSE  TRUE
##   neg 15010   615
##   pos    21   354
```

```
precision<-(cm[1,1]+cm[2,2])/sum(cm);precision
```

```
## [1] 0.96025
```

```
## Coste Total
coste_total_log <- cm[1,2]*10 + cm[2,1]*500
coste_total_log
```

```
## [1] 16650
```

```
# AUC value
auc_ROCR <- performance(ROCRpred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]];auc_ROCR
```

```
## [1] 0.9875581
```

```
train_log<-NULL
test_log<-NULL
```

5.5 Naive Bayes

El último algoritmo implantado es Naive Bayes. El paquete NaiveBayes proporciona un algoritmo eficiente para la implantación de Naive Bayes. Por ejemplo, se han realizado pruebas con el Naive Bayes del paquete caret, oteniendo unos tiempos superiores a 10 minutos, dado al gran tamaño de la muestra. En cambio con el paquete NaiveBayes es mucho más rápido.Se ha aplicado Naive Bayes a los datos imputados por k-means y con transformación Box-Cox. Se ha realizado el balanceado de la muestra mediante ROSE.

La precisión del modelo es 94.35%. Hay 39 falsos negativos. El coste total basado en la matriz de confusión es 28135, also peor al obtenido a la regresiín logística balanceado con SMOTE.Hay que destacar la rapidez de la ejecución.

```
if (!require("e1071")) install.packages("e1071")
```

```
## Loading required package: e1071
```

```
##
## Attaching package: 'e1071'
```

```
## The following objects are masked from 'package:dlookr':
##
##     kurtosis, skewness
```

```
## The following object is masked from 'package:imputeMissings':
##
##     impute
```

```
library(e1071) # Naive Bayes en este paquete
if (!require("ROSE")) install.packages("ROSE")
```

```
## Loading required package: ROSE
```

```
## Loaded ROSE 0.0-3
```

```
library(ROSE) # ROSE para balanceado


# ModeloS
train_balanced <- ROSE(class ~ ., data=train_bc_km)$data
train_balanced$class<-as.factor(train_balanced$class)
test_nb<-test_bc_km
test_nb$class<-as.factor(test_nb$class)

model_nb <- naiveBayes(class ~ ., data = train_balanced)
#model_nb
summary(model_nb)
```

```
##           Length Class  Mode
## apriori     2    table  numeric
## tables    128    -none- list
## levels      2    -none- character
## isnumeric 128    -none- logical
## call        4    -none- call
```

```
## Predicción de valores
prediction <- predict(model_nb, as.data.frame(test_nb))

## Matriz de confusión
cm <- table(test_nb$class,prediction)
cm
```

```
##      prediction
##         neg   pos
##   neg 14760   865
##   pos    39   336
```

```
cm_stat<-confusionMatrix(cm,positive="pos")
cm_stat
```

```
## Confusion Matrix and Statistics
##
##      prediction
##         neg   pos
##   neg 14760   865
##   pos    39   336
##
##                Accuracy : 0.9435
##                  95% CI : (0.9398, 0.947)
##     No Information Rate : 0.9249
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4051
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.27977
##             Specificity : 0.99736
##          Pos Pred Value : 0.89600
##          Neg Pred Value : 0.94464
##              Prevalence : 0.07506
##          Detection Rate : 0.02100
##    Detection Prevalence : 0.02344
##       Balanced Accuracy : 0.63857
##
##        'Positive' Class : pos
##
```

```
precision<-(cm[1,1]+cm[2,2])/sum(cm);precision
```

```
## [1] 0.9435
```

```
## Coste Total
coste_total_nb <- cm[1,2]*10 + cm[2,1]*500
coste_total_nb
```

```
## [1] 28150
```

5.6 Resumen de resultados

El mejor resultado se ha obtenido con la regresión logística, utilizando la técnica SMOTE para balancear los datos. Si bien los resultados han sido mejores con el balanceo por pesos, el modelo resultado ha obtenido 0 variables significativas, lo que no es admisible. Cabe destacar cómo han reducido los falsos negativos al balancear los datos con SMOTE, ya que el problema de los algorimtos de clasificación es que tienden a favorecer la clase mayoritaria (si no se utiliza otra técnica para balancear casos).

Se resumen en la tabla los resultados obtenidos en los diferentes análisis:

Modelo	Imputación	Transformación	Balanceo	Precisión	Falso Neg.	Coste Total	Reglas/Var.Sig	Comentario
c5.0	k-means	Datos PCA	No	97.24%	267 (71.2%)	135240	39 reglas	
c5.0	mediana	Datos PCA	No	97.39%	276 (73.5%)	127300	50 reglas	
Reg.Log	k-means	No	No	98.73%	139 (37.0%)	70135	49 var. sig.	
Reg.Log	k-means	Box-Cox	No	98.75%	132 (35.2%)	66670	42 var. sig	
Reg.Log	k-means	Box-Cox	SMOTE	95.94%	32 (8.5%	22660	48 var. sig	
Reg.Log	k-means	Box-Cox	Weights	97.70%	23 (6.1%)	17770	0!!!	0 var.sig.!!
Reg.Log	mediana	No	No	98.80%	133 (35.4%)	66960	37 var. sig.	
Reg.Log	mediana	Box-Cox	No	98.93%	127 (35.2%)	63940	51 var. sig	
Reg.Log	mediana	Box-Cox	SMOTE	96.26%	22 (8.5%	16980	55 var. sig	
Reg.Log	mediana	Box-Cox	Weights	98.35%	21 (5.6%)	16650	0!!!	0 var.sig.!!
Naive Bayes	k-means	Box-Cox	ROSE (50%)	94.35%	39 (10.4%)	28150		

6 Conclusiones

El objetivo de esta práctica ha sido profundizar en las técnicas de limpieza y análisis de datos median un caso de uso. El conjunto de datos seleccionado ha sido el **APS Failure at Scania Trucks**. El objetivo final de trabajo es predecir con los datos proporcionados si un fallo puede deberse al sistema APS. Se proporciona un conjunto de datos de entrenamiento (60000 filas y 171 atributos) y testeo (16000 filas y 171 atributos). Todas las variables están anonimizadas. El conjunto de datos está desequilibrado entre los negativos (98.88%) y positivos (1.12%).

El trabajo ha comenzado con la limpieza de datos y tratamiento de perdidos y ceros. El conjunto de datos de entrenamiento tiene un tamaño elevado, por lo que se han utilizado técnicas exploratorias para adquirir de forma ágil información de todas las variables. El conjunto de datos original incluía un alto número de perdidos (169 columnas de 171 presentaban algún valor perdido). Se han eliminado columnas con perdidos superiores al 20% (24 variables). Para el resto de datos perdidos, se ha optado por la función ClustImpute (función ClustImpute) que realiza una función por cluster k-means. También se ha utilizado la imputación más habitual por la mediana y comparado más adelante en la fase de análisis los resultados de ambas imputaciones. Como se ha comentado, no ha resultado posible la imputación con missForest, debido al gran tamaño de la muestra (se ha realizado un ejemplo con una muestra menor con resultado satisfactorio). Para terminar con la fase de preprocesamiento, se ha utilizado también el Análisis de Componentes Principales, sí que se ha obtenido una reducción considerable (50% de variables para explicar el 95% de la varianza) del número de variables, aunque el número de variables sigue siendo elevado. Los datos no presentaban una distribución normal. Se ha optado por una transformación Box-Cox logística (lamda=0) para mejorar su semejanza en la normal, pero en los casos de muchos ceros la distribución ha seguido teniendo una distribución diferente a la normal.

En la fase de análisis, se han utilizado el árbol de decisión c5.0, regresión logística y Naive Bayes, en diferentes escenarios. El mejor resultado se ha obtenido con la regresión logística, utilizando la técnica SMOTE para balancear los datos. Cabe destacar cómo se han reducido los falsos negativos al balancear los datos con SMOTE, ya que el problema de los algorimtos de clasificación es que tienden a favorecer la clase mayoritaria (si no se utiliza otra técnica para balancear casos). En el caso de Naive Bayes se ha utilizado el paquete ROSE para el balanceo de datos.

El mejor coste total del modelo de predicción ha resultado ser con el modelo de regresión logística con SMOTE, sobre datos imputados con la mediana, obteniéndose un coste total de 16980.

Por último, el principal reto de este conjunto de datos de SCANIA ha sido el gran tamaño del mismo (tanto número de filas como de columnas), seguido del desequilibrio entre las clases positivo y negativa. Para trabajar con grandes volúmenes de datos, es importante utilizar técnicas exploratorias (EDA, Exploratory Data Analysis) que nos permitan obtener información general de todas las variables de un “vistazo”, así como utilizar técnicas de análisis eficientes computacionalmente. Durante el trabajo hemos descubierto paquetes como skimr, ClustImpute, dlookr o NaiveBayes, que han facilitado nuestro análisis. El otro reto ha sido el desequilibrio entre las clases positivas y negativas, donde se han descubierto técnicas como SMOTE y ROSE que solucionan el problema mediantes upsampling, downsampling o “mixto”.

7 Contribuciones

Los autores de este trabajo Iván Jiménez (IJ) e Itziar Ricondo (IR) han contribuido en las diferentes fases del trabajo, como consta en las siguientes firmas.

Contribuciones	Firma
Investigación previa	IJ,IR
Redacción de las respuestas	IJ,IR
Desarrollo de código	IJ,IR

8 Referencias

[1] https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks (https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks)

[2] https://www.r-bloggers.com/2019/06/how-data-tables-fread-can-save-you-a-lot-of-time-and-memory-and-take-input-from-shell-commands/ (https://www.r-bloggers.com/2019/06/how-data-tables-fread-can-save-you-a-lot-of-time-and-memory-and-take-input-from-shell-commands/)

[3] https://www.datanovia.com/en/blog/display-a-beautiful-summary-statistics-in-r-using-skimr-package/ (https://www.datanovia.com/en/blog/display-a-beautiful-summary-statistics-in-r-using-skimr-package/)

[4] https://www.r-bloggers.com/2019/06/intoducing-clustimpute-a-new-approach-for-k-means-clustering-with-build-in-missing-data-imputation/ (https://www.r-bloggers.com/2019/06/intoducing-clustimpute-a-new-approach-for-k-means-clustering-with-build-in-missing-data-imputation/)

[5] https://rpubs.com/lmorgan95/MissForest (https://rpubs.com/lmorgan95/MissForest)

[6] https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/SMOTE (https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/SMOTE)

[7] https://rdrr.io/cran/naivebayes/ (https://rdrr.io/cran/naivebayes/)