Práctica 1
Sistemas Inteligentes para la Gestión en la Empresa

Pre-procesamiento de datos y clasificación binaria

18 de abril de 2019

Juan Carlos Serrano Pérez - jcsp0003@correo.ugr.es

Índice

Introducción

Se trabajará sobre el conjunto de datos de préstamos proporcionado junto a este guión, que es una variación del ofrecido en la competición de Kaggle Santander Customer Transaction Prediction.

https://www.kaggle.com/c/santander-customer-transaction-prediction

La descripción de las variables de este conjunto de datos se encuentra en la sección DATA de esta misma web.

El problema consiste en predecir si un cliente realizará una transacción en el futuro (target) a partir del resto de variables (200). Trataremos el conjunto de datos como un problema de
clasificación binaria, con dos posibles salidas: {Yes, No}.

Exploración

Para comenzar la exploración de los datos comenzaremos utilizando los comandos summary y df_status para obtener un resumen general de los datos:

```
#Obtengo un resumen de los datos
summary(datos)

#Obtengo estadisticas como el numero de ceros o de N/A
status <- df_status(datos)
```

	variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
1	ID_code	0	0.00	0	0.00	0	0	factor	200000
2	target	181989	90.99	0	0.00	0	0	integer	2
3	var_0	0	0.00	17	0.01	0	0	numeric	94670
4	var_1	1	0.00	11	0.01	0	0	numeric	108931
5	var_2	0	0.00	19	0.01	0	0	numeric	86554
6	var_3	0	0.00	16	0.01	0	0	numeric	74593
7	var_4	0	0.00	22	0.01	0	0	numeric	63513
8	var_5	3	0.00	21	0.01	0	0	numeric	141017
9	var_6	0	0.00	21	0.01	0	0	numeric	38599
10	var_7	0	0.00	20	0.01	0	0	numeric	103059
11	var_8	1	0.00	17	0.01	0	0	numeric	98615
12	var_9	0	0.00	19	0.01	0	0	numeric	49417
13	var_10	1	0.00	12	0.01	0	0	numeric	128760
14	var_11	2	0.00	24	0.01	0	0	numeric	130183
15	var_12	0	0.00	26	0.01	0	0	numeric	9560
16	var_13	0	0.00	20	0.01	0	0	numeric	115175
17	var_14	0	0.00	20	0.01	0	0	numeric	79119
18	var_15	0	0.00	20	0.01	0	0	numeric	19810
19	var_16	0	0.00	21	0.01	0	0	numeric	86914
20	var_17	2	0.00	18	0.01	0	0	numeric	137819

A partir de lo anterior obtenemos información interesante como:

- El conjunto está compuesto por 200000 filas y 202 columnas.
- Las variables ID_code y target son de tipo factor e integer respectivamente, mientras que el resto son de tipo numeric.
- Los valores de ID_code con únicos y sirven como identificador por lo que no nos serán útiles para la clasificación y podemos eliminar la columna.
- La variable target solo toma 2 valores (0 o 1).
- El número de datos NA no supera el 0.02% en ningún caso, por lo que no se considerará el eliminar columnas enteras por falta de datos.

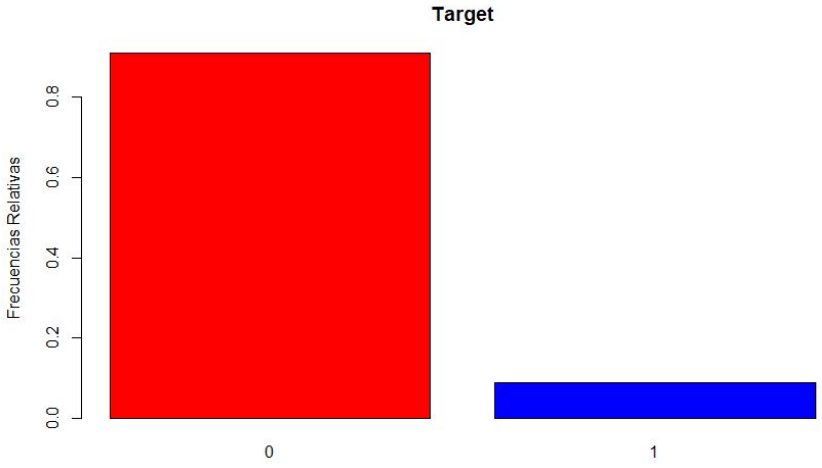
El objetivo del proyecto es el de predecir si un cliente realizará en un futuro una transacción, el atributo target, así que lo estudiaremos para ver qué valores toma y la distribución de los datos.

Como se comentó anteriormente, target solo toma dos valores que como vemos son el ‘0’ y el ‘1’, entre los cuales hay una gran descompensación en la distribución de los datos, siendo el primero en el 90.99% de los casos frente al 9% del otro.

```
> prop.table(table(train$target))

       0        1
0.909945 0.090055
```

Pre-procesamiento

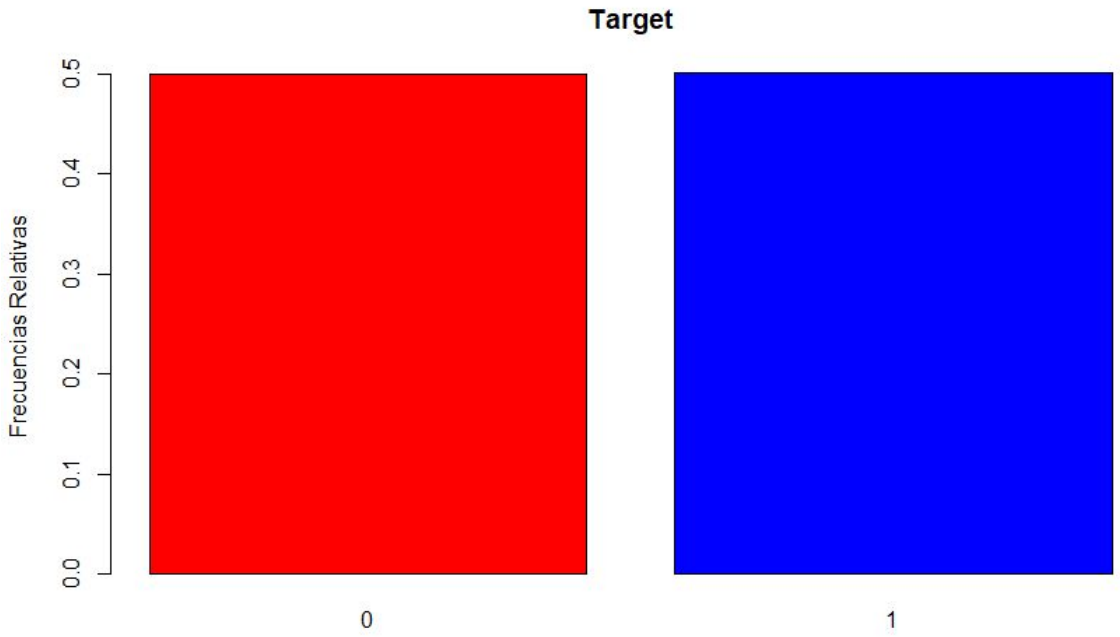
Como se comentó hay una gran descompensación en el balenceo de los datos de la variable target como podemos apreciar de forma mucho más visual en el siguiente diagrama de barras donde se representa ambos valores de forma independiente frente a su frecuencia relativa.



Debido a la descompensación entre los dos valores que toma la variable Target se va a proceder a su balanceo. Haciendo uso de la función *ovun.sample* dotada con técnicas de resampling se determina muestras mediante submuestreo sin reemplazamiento de la clase mayoritaria hasta que el número de N se determinado o bien “p”.

```
> #Balanceo los datos
> table(datos$target)

     0      1
181989  18011
> datos <- ovun.sample(target ~., data=datos, p=0.5, seed=1, method="under")$data
> table(datos$target)

    0     1
17602 17664
```

De esta forma, lo volvemos a representar y vemos que los datos están correctamente balanceados.



Una vez balanceados los datos, se comenzará eliminando el atributo ID_code debido a que es un valor único que sirve como identificador, por lo que no nos es útil para la predicción, y posteriormente se eliminarán las filas con valores NA.

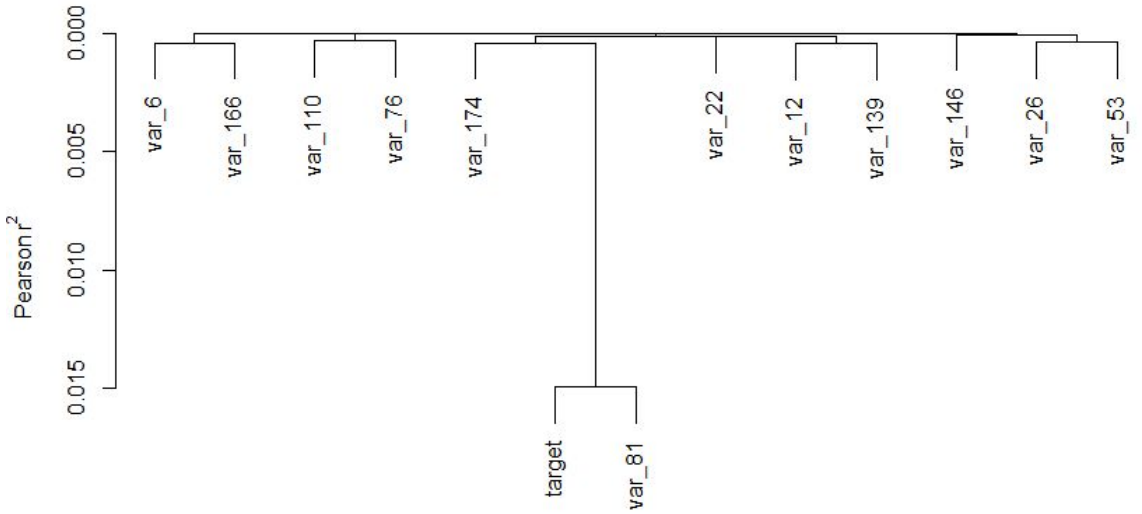
```
#Elimino la columna ID_code
datos <- select(datos, -ID_code)

#Elimino las entradas con valores NA
datos <- na.omit(datos)
```

Para el estudio de la correlación de las variables va a realizar mediante Pearson para así reducir el número de variables y obtener las más importantes.

```
cor_target <- correlation_table(data_num, target='target')
important_vars <- cor_target %>%
  filter(abs(target) >= 0.1)

datos <- datos %>%
  select(one_of(important_vars$Variable))

# Alta correlacion entre si
data_num <- datos %>%
  na.exclude() %>%
  mutate_if(is.character, as.factor) %>%
```

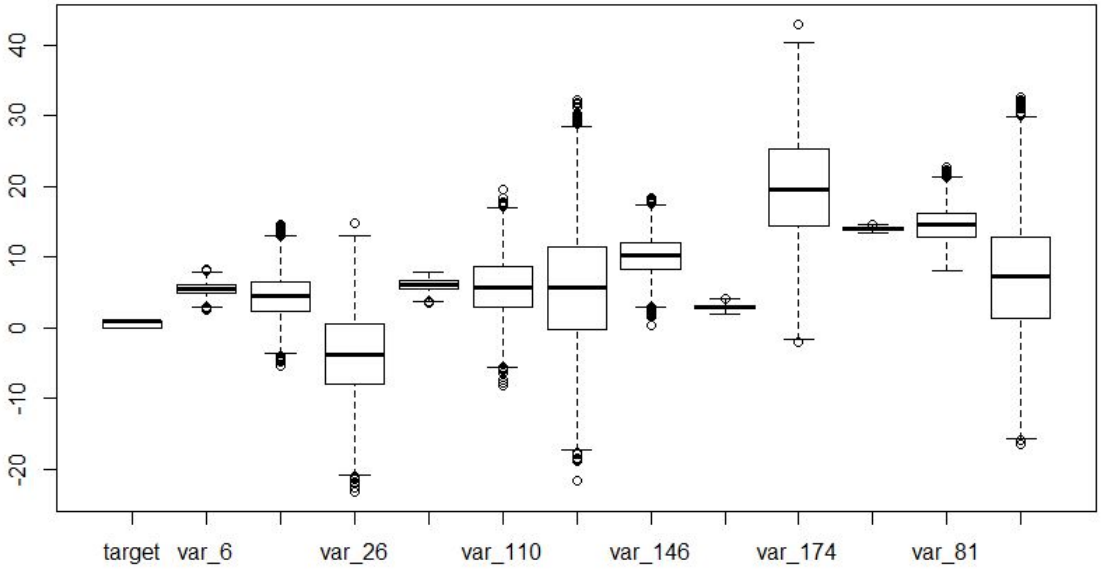
```
  mutate_if(is.factor, as.numeric)
rcorr_result <- rcorr(as.matrix(data_num))
cor_matrix <- as.tibble(rcorr_result$r, rownames = "variable")
corrplot(rcorr_result$r, type = "upper", order = "original", tl.col =
"black", tl.srt = 45)

v <- varclus(as.matrix(data_num), similarity="pearson")
plot(v)
```

A partir de lo anterior hemos reducido el conjunto a 13 variables que podemos ver su relación en el siguiente diagrama.

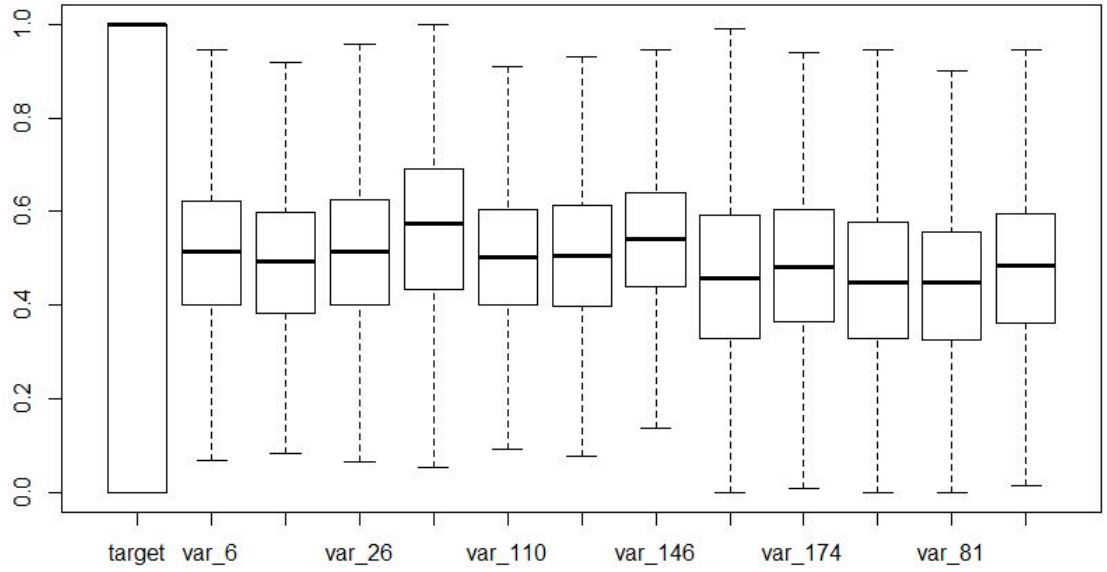


Para el estudio de la dispersión de los datos se realizará un diagrama de cajas para observar la dispersión de todas las variables.

```
boxplot(datos)
```



En el diagrama podemos ver la dispersión de todas las variables, en este punto podemos considerar dos acciones viendo los datos: su normalización y la eliminación de los outliers. Tras lo cual el diagrama quedará de la siguiente forma:



Debido al tamaño del conjunto de datos y las bajas prestaciones del equipo no ha podido aplicar el tratamiento del ruido de los datos.

Clasificación

Para la clasificación se ha dividido el conjunto de datos ya pre-procesados en dos subconjuntos.

```
# Conjuntos de entrenamiento y validación
trainIndex <-
  createDataPartition(datos$target,
                      p = .8,
                      list = FALSE,
                      times = 1)
train <- datos[trainIndex, ]
val   <- datos[-trainIndex, ]
```

La partición se ha realizado dándole un 80% de los datos al conjunto de entrenamiento y el 20% restante al conjunto de validación.

La precisión media y el tiempo de ejecución de los algoritmos utilizados se muestra en la siguiente tabla (*en el caso de SVM se ejecutó sobre un subconjunto menor de lo anteriormente explicado, un 30% de los datos, debido largo tiempo de ejecución):

	Accuracy	**Tiempo**
RPART	0.5883	11.83264 secs
RF	0.6632	43.11453 mins
SVM*	0.6969	1.024036 hours

Como se ve en los resultados de la tabla, el algoritmo que nos ha dado peor resultado medio de clasificación es RPART pero en un tiempo muy corto. Mientras que SVM da el mejor resultado de predicción pero es el que conlleva mayor tiempo de ejecución (aún después de reducir el conjunto de datos en un 70%).

En el caso de querer unos resultados aceptables en un buen tiempo, lo correspondiente sería utilizar el método RF ya que da resultados similares al anterior pero en un tiempo de ejecución mucho menor.

Conclusiones

Para ser la primera toma de contacto con el lenguaje R y en la clasificación binaria de datos se considera que aunque el 69,69% de precisión, aún quedándose lejos del primer puesto de Kaggle es muy aceptable.

La estructura del conjunto de datos utilizado no ha sido de gran ayuda para una primera toma de contacto en la clasificación debido a tu gran número de variables abstractas que no conocíamos su significado y de filas que ha afectado en gran medida a los tiempos de ejecución y limitando la técnicas que podíamos utilizar por falta de memoria.

En último lugar hablar de la importancia del pre-procesamiento de los datos realizado. Debido a que si omitimos las técnicas utilizadas antes de comenzar a clasificar como el balanceo de los datos, eliminación de valores perdidos o de columnas con poca información, no solo se incrementa en gran medida en tiempo de entrenamiento del modelo, sino tendríamos cerca de un 91% de acierto debido a un sobreaprendizaje de la variable target con valor 0, y en caso de ponerlo en práctica con el conjunto de test de Kaggle tan solo obtenemos un 50% de acierto. Debido a esto, queda visto a que gracias al pre-procesamiento hemos reducido en gran medida en tiempo de ejecución y un 20% de acierto adicional.

Bibliografía

[1] Diego Calvo. (2019). Eliminar NA o valores nulos en R - Diego Calvo. [online] Available at: http://www.diegocalvo.es/eliminar-na-o-valores-nulos-en-r/ [Accessed 18 Apr. 2019].

[2] GitHub. (2019). jgromero/sige2019. [online] Available at: https://github.com/jgromero/sige2019/blob/master/pr%C3%A1cticas/02.%20Depuracion%20y%20calidad%20de%20datos/titanic/titanic.Rmd [Accessed 18 Apr. 2019].