

Taller: Cómo Crear y productivizar modelos de CS

Nov 2023

FRANCISCO J. RODRÍGUEZ ARAGÓN

SCORE MANAGER at ONEY

Ph D in Statistic by Cordoba Univ.

Associate Professional Risk Manager Certified by APRM

Operational Risk Manager Certified by APRM



Summary

Introducción: un poco de mi actividad y de mí mismo

1. ¿Qué es un modelo de credit scoring?
2. DEMO de la librería *scorecard*
3. El ML ¿Son los padres?
4. DEMO Scorecards bajo el paradigma ML
5. Conclusiones finales

INTRODUCCIÓN

¿Qué es Oney?

¿Qué es Oney?

Especialista en medios de pago y financiación para el retail y el consumidor final

- 12 países
- 8M de clientes
- 35 años acompañando a los retailers
- +600 partners en Europa

Tarjetas

Pago Aplazado 3x 4x

E-financiación 6x 10x 12x

Crédito al consumo

Seguros





PARTE 1

¿Qué es un modelo de Credit Scoring?

one



oney | 5

Los modelos más utilizados en la actualidad

- En esencia los modelos de Credit Scoring no son más que una reformulación de un modelo de regresión logística subyacente
- Son modelos poco conocidos por los Data Scientist actuales a pesar de su gran aplicación actual
- Los modelos de credit scoring están en:
 - Cada compra que se realiza con una tarjeta de crédito, aunque aquí poco a poco están llegando (que no imponiéndose) los modelos de Machine Learning
 - En la concesión de hipotecas y en general en todo tipo de préstamos al consumo ya que el regulador bancario prácticamente obliga a las entidades a usar este tipo de modelos en sus créditos

En España, las últimas cifras del **Banco de España** reflejan esta tendencia al alza. Por un lado, el saldo de créditos al consumo alcanzó casi los 188.000 millones de euros en junio, **un 5% más que a principios de año**. Si bien hay un pico cíclico en esta fecha, por todos los créditos que se piden en junio para contratar vacaciones, este auge se apoya en otras dos cifras.

188.000 mill. De euros al menos gestionados por estos modelos en el sector revolving!!!!

RIESGOS

Alarma bancaria por un 'boom' de créditos al consumo para combatir la inflación

La financiación al consumo crece un 5% y el saldo en tarjetas 'revolving' alcanza máximos de dos años y medio. Mientras, la morosidad de las financieras toca el pico desde 2016

Por **Jorge Zuloaga**

03/08/2022 - 05:00

El objetivo básico de los modelos de CS

- Se trata de medir el grado de ocurrencia de una variable binaria tipo 0 - 1
- Habitualmente en banca, esta variable binaria será el pago/impago de una obligación de crédito a un año vista, por lo que a esta probabilidad se la denomina *PD* o *probabilidad de Default*
- En un modelo de CS se culmina con la construcción de una targeta de puntuación como se verá a continuación, pero previamente existe una regresión logística subyacente

$$P[Y = 1] = \frac{1}{1 + e^{\alpha_0 + \alpha_1 WoE_Age}}$$

- Además se va a exigir un alto grado de interpretación y de estabilidad:
 - Estabilidad a nivel de tramos de puntuaciones de score totales
 - Estabilidad a nivel de tramos de puntuaciones de variables o score parciales
 - Las variables deben tener lógica de negocio
 - Las variables deben de ser significativas y discriminantes en el modelo

Un modelo “Tarjeta de Puntuación”

Variable	Tramo	Tasa mora	WOE	Score	Beta	IV	SCR
Pasivo	-100	31,46%	-0,797	36	-0,4032	0,2775	10,99%
	100-800	17,90%	-0,053	45			
	800-2000	18,23%	0,305	49			
	2000-3700	11,10%	0,504	51			
	3700+	1,11%	0,992	57			
Triad	-650	22,74%	-0,353	39	-0,6724	0,2667	35,91%
	650-675	15,21%	0,136	48			
	675-700	9,11%	0,724	59			
	700+	1,60%	1,456	74			
Ratio_AC_PAS	1	34,18%	-0,921	36	-0,3718	0,1871	5,16%
	2	22,34%	-0,330	42			
	3	16,89%	0,017	46			
	4	12,98%	0,537	49			
Nacionalidad	1	27,48%	-0,335	33	-0,7051	0,1345	18,66%
	2	22,37%	-0,110	39			
	3	18,75%	-0,110	40			
	4	16,50%	0,046	45			
	5	13,25%	0,303	53			
	6	7,50%	0,937	58			
Incidencias	CON INCIDENCIAS	27,40%	-0,601	34	-0,5171	0,1112	6,56%
	SIN INCIDENCIAS	14,53%	0,196	45			
Antigüedad	-11	19,61%	-0,165	43	-0,6715	0,0802	9,92%
	11-29	18,84%	-0,116	43			
	29-53	14,21%	0,222	50			
	53+	9,60%	0,666	58			
Profesion	1	22,78%	-0,355	40	-0,4904	0,0432	2,92%
	2	19,20%	-0,139	43			
	3	16,19%	0,068	46			
	4	14,78%	0,176	48			
	5	9,95%	0,627	54			
Antig_Emp	-12	19,36%	-0,149	43	-0,6211	0,0419	3,75%
	12-36	15,28%	0,137	48			
	36+	11,80%	0,436	53			
Est_Civil	1	19,52%	-0,159	42	-0,7968	0,0351	4,80%
	2	17,41%	-0,019	45			
	3	14,32%	0,213	50			
	4	12,33%	0,386	54			
Provincia	1	20,26%	-0,206	42	-0,6165	0,0319	3,13%
	2	18,05%	-0,063	44			
	3	16,53%	0,044	46			
	4	13,64%	0,270	50			
	5	11,45%	0,469	54			

En los scoring las variables deben ser trameadas ¿Cómo se consigue esto?

Un modelo de Credit Scoring o de tarjeta de puntuación es un algoritmo donde a cada variable de entre una colección, se les asocia, en función de su valor, una puntuación denominada score parcial.

Nótese qué, en una regresión logística, se tiene una ecuación matemática donde existe una exponencial, en cambio en un CS lo anterior se transforma a una simple suma de puntuaciones !!!

Fuente de la scorecard: <https://docplayer.es/49254028-Desarrollo-y-validacion-de-modelo-de-scoring-de-admision-para-tarjetas-de-credito-con-metodologia-de-inferencia-de-denegados.html>

En los scoring se generan puntuaciones que pueden ser algebraicamente sumadas para cada cliente ¿Cómo se consigue esto?

Building blocks en un CS

El resultado final de un Credit Scoring procede de la suma aritmética de unos Scoring Parciales:

- Cada variable genera un score parcial
- Los score parciales aunque valores numéricos, están discretizados y su rango de variación es mayor o menor en función de cómo separen la tasa de la variable objetivo

¿Por qué se usa una regresión subyacente en los CS y no otro tipo de modelo?

Es justamente el tratamiento de esta cuestión y sus consecuencias, lo que se va analiza en esta ponencia

Un modelo muy interpretable

- La propiedad esencial de un modelo de CS es su interpretabilidad, sin embargo ¿Qué sucede cuando se compara con un modelo de ML
- ¿Por qué un CS frente a un Random Forest si este último es más predictivo?

EBA DISCUSSION PAPER ON MACHINE LEARNING FOR IRB MODELS

EBA/DP/2021/04

11 NOVEMBER 2021

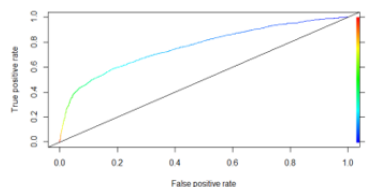
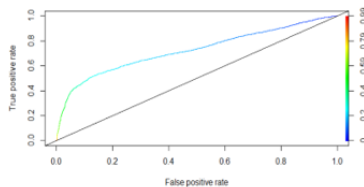
En este caso particular se observa como un Random Forest mejora la predictividad desde un 0.7238 de una regresión logística hasta un 0.7633, es decir, algo más de un 5.45%

Fuente: <https://github.com/FJROAR/Ejemplo-Blog-RWA>

```
***** Importancia de variables Voting_Classifier *****
Logit Random_Forest
Variables
log_providerA_score 0.376384 0.366804
providerZ_score 0.002200 0.231568
order_amount 0.000355 0.108292
num_instalments_initial 0.030441 0.099995
dum_cellular_ip_address 0.368038 0.051138
dum_wifi_ip_userType -0.353291 0.023141
dum_traveler_ip_address 0.000000 0.000000
pm_card_expiration_year -0.000051 0.016783
providerY_score -0.231073 0.044099
dum_hosting_ip_address 0.000000 0.000000
dum_master_method_card_type 0.011679 0.005051
dum_extranjero_pm_card_country_code 0.000000 0.000870
num_edad -0.005074 0.040019
device_cookies_enabled 0.020754 0.003667
dum_amex_method_card_type 0.000000 0.000000
dum_business_ip_address 0.000000 0.000326
customer_visit -0.020313 0.008013
dum_satellite_ip_userType 0.000000 0.000000
ip_reputation -0.481478 0.000235
```

Regresión Logística: 72.38%

Random Forest: 76.33%



Regresión Logística	Random Forest
Modelo Estándar: 317.625.000	Modelo Estándar: 306.547.500
Modelo Avanzado: 368.339.934	Modelo Avanzado: 300.048.403

PARTE 2

DEMO DE LA LIBRERÍA scorecard



Funciones interesantes

- Una librería muy completa que cubre todo el ciclo de modelización

R topics documented:

describe	2
gains_table	3
germancredit	4
iv	5
one_hot	6
perf_cv	7
perf_eva	9
perf_psi	10
replace_na	12
report	13
scorecard	15
scorecard2	17
scorecard_ply	18
scorecard_pmml	19
split_df	20
var_filter	21
var_scale	22
vif	23
woebin	24
woebin_adj	27
woebin_plot	28
woebin_ply	30

Pre-requisitos

- Para seguir el taller es necesario descargarse el siguiente conjunto de datos:

<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/>


- Créese una estructura lógica de carpetas
- Conviene además tener instalado Rstudio e instalarse las siguientes librerías:

dplyr
data.table
scorecard
scorecardModelUtils
plotly

Datos y entorno de trabajo

- En general hay mucho que hacer para preparar la información no obstante, aquí se parte de algo que en general es totalmente irreal, un conjunto de datos con sus variables prácticamente todas prepradas
- No nos interesa tener un alto grado de predictividad, sino:
 - Entender los fundamentos de un modelo de Credit Scoring y cómo se construye
 - Identificar los elementos claves de su estructura
 - Interpretación y diagnosis
 - Serialización y pautas para una puesta en producción sencilla del modelo que se haga
- Para ello se toma como *entorno de trabajo* la estructura de carpetas creada y se ejecuta el siguiente código de R que se explica paso a paso:

FORMACION > TALLER_RMADRID2024			
Nombre	Fecha de modificación	Tipo	Tamaño
code	30/11/2023 22:22	Carpeta de archivos	
data	30/11/2023 21:03	Carpeta de archivos	
doc	03/12/2023 18:04	Carpeta de archivos	

 00_preparacion_bc

Desarrollo de la demo CS con R

- La librería *scorecard* fue desarrollada para *R* y actualmente Python ha copiado su funcionalidad también, por lo que existe en los 2 lenguajes. A pesar de la importancia de estos modelos, esta librería es bastante reciente y se puso disponible alrededor del 2019. Su uso en R ahorra muchas horas de trabajo en la construcción de estos modelos
- Paso 1: Lectura de la información

```
#https://cran.r-project.org/web/packages/scorecard/vignettes/demo.html  
  
library(data.table)  
library(dplyr)  
library(scorecard)  
  
df <- fread("data/creditcard.csv", sep = ",")  
  
names(df)
```

Data	
df	284807 obs. of 31 variables

Desarrollo de la demo CS con R

- Paso 2: Limpieza de datos (adicional) y análisis de las variables (por tiempo, se elude este paso ya que los datos están muy limpios, en la realidad aquí debe usarse todo el tiempo necesario)
- Paso 3: Separate sample (hasta aquí, no hay nada distinto respecto a un modelo de Machine Learning actual)

```
df_list <- split_df(df, y = "Class", ratios = c(0.6, 0.4), seed = 30)
label_list <- lapply(df_list, function(x) x$Class)

df_train <- df_list$train
df_test <- df_list$test
```


Desarrollo de la demo CS con R

- Paso 4: TRAMEADO DE VARIABLES. En muchos modelos de ML este paso no se da y no es obligatorio, en cambio en los modelos CS es obligatorio darlo

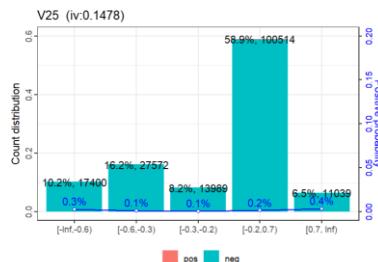
¿Y por qué no tenerlo en cuenta también en los modelos de ML? ¿Los mejoraría?

```
bins_train <- woebin(df_train, y = "Class")
#Ejemplo de un bin:
#bins_train$Time
```

variable	bin	count	count_distr	neg	pos	posprob	woe	bin_iv	total_iv	breaks	is_special_values
Time	[-Inf,30000)	11395	0.06682736	11346	49	0.0043001316	0.923304704	9.339604e-02	0.3027489	30000	FALSE
Time	[30000,40000)	12553	0.07361859	12543	10	0.0007966223	-0.766227983	3.021969e-02	0.3027489	40000	FALSE
Time	[40000,85000)	61728	0.36201133	61623	105	0.0017010109	-0.006725162	1.631823e-05	0.3027489	85000	FALSE
Time	[85000,110000)	8813	0.05168491	8772	41	0.0046522183	1.002356901	8.908773e-02	0.3027489	110000	FALSE
Time	[110000,125000)	14296	0.08384062	14288	8	0.0005595971	-1.119628817	6.330402e-02	0.3027489	125000	FALSE
Time	[125000, Inf)	61729	0.36201720	61650	79	0.0012797875	-0.291675712	2.672512e-02	0.3027489	Inf	FALSE

La visualización de estos tramos es siempre relevante y es donde se focaliza gran parte de la construcción de modelos

```
#Plotting bins
woebin_plot(bins_train)
```



Desarrollo de la demo CS con R

- Paso 4: TRAMEADO DE VARIABLES: Características esenciales del trameado

El paso del trameado es esencial ya que:

- Introduce y mide las **tendencias esperadas**
- Los tramos deben ser **representativos**
- Los tramos deben tener **SENTIDO DE NEGOCIO**

¿Qué significa sentido de negocio? ¿Lo dejamos todo en manos de un algoritmo? ¿Manipulamos datos?

Desarrollo de la demo CS con R

- Paso 5: TRANSFORMACIÓN WoE: Todas las variables son transformadas bajo la siguiente fórmula

$$WoE = \ln \left(\frac{\% \text{ de no eventos}}{\% \text{ de eventos}} \right)$$

- % de no eventos: La proporción de no eventos (por ejemplo, clientes que no incumplen en el pago) en una categoría.
- % de eventos: La proporción de eventos (por ejemplo, clientes que incumplen en el pago) en una categoría.

¿Qué significa un WoE positivo? ¿Y negativo?

Desarrollo de la demo CS con R

- Paso 5: TRANSFORMACIÓN WoE: Una medida de la importancia univariante el IV

$$IV = \sum_{i=1}^n (\% \text{ de no eventos}_i - \% \text{ de eventos}_i) \times \text{WoE}_i$$

n : Número de categorías en la variable.

% de no eventos _{i} : La proporción de no eventos en la categoría i .

% de eventos _{i} : La proporción de eventos en la categoría i .

WoE _{i} : El Weight of Evidence para la categoría i .

0 a 0.02: Variable predictiva insignificante.

0.02 a 0.1: Variable predictiva débil.

0.1 a 0.3: Variable predictiva moderada.

0.3 a 0.5: Variable predictiva fuerte.

0.5: Variable altamente predictiva.

Range	Bins	Non events	Events	% of Non-Events	% of Events	WOE	IV
0-50	1	197	20	5.4%	5.9%	-0.0952	0.0005
51-100	2	450	34	12.3%	10.1%	0.2002	0.0045
101-150	3	492	39	13.4%	11.5%	0.1522	0.0029
151-200	4	597	51	16.3%	15.1%	0.0774	0.0009
201-250	5	609	54	16.6%	16.0%	0.0401	0.0003
251-300	6	582	55	15.9%	16.3%	-0.0236	0.0001
301-350	7	386	41	10.5%	12.1%	-0.1405	0.0022
351-400	8	165	23	4.5%	6.8%	-0.4123	0.0095
>401	9	184	21	5.0%	6.2%	-0.2123	0.0025
Total		3662	338				0.0234

¿Por qué es siempre positivo el IV?

Desarrollo de la demo CS con R

- Paso 6: Construcción de un modelo de regresión logística subyacente con las variables trameadas (training del modelo)

```
dt_woe_list = lapply(df_list, function(x) woebin_ply(x, bins_train))
m1 = glm( Class ~ ., family = binomial(), data = dt_woe_list$train)

summary(m1)
```

```
> summary(m1)

Call:
glm(formula = Class ~ ., family = binomial(), data = dt_woe_list$train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4489  -0.0153  -0.0084  -0.0051   4.6185

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.117497    0.131948  -38.784  < 2e-16 ***
Time_woe    -0.328063    0.189628  -1.730  0.083624 .
V1_woe       0.221128    0.119524   1.850  0.064303 .
V2_woe      -0.326117    0.093742  -3.479  0.000504 ***
```

¿Cómo escoger un modelo final? ¿Cuál es la fórmula del modelo construido?

Desarrollo de la demo CS con R

- Paso 7: Evaluación del modelo (habitual en ML también)

```
pred_list = lapply(dt_woe_list, function(x) predict(m1, x, type='response'))  
perf = perf_eva(pred = pred_list, label = label_list, show_plot = c("ks", "roc"),  
               pred_desc = F)
```

¿Cómo analizaría si el modelo creado es o no estable ante cambios de muestras?

¿Aparte de la ROC qué otras medidas usáis en vuestros modelos de ML?

Desarrollo de la demo CS con R

- Paso 8: TARJETA DE PUNTUACIÓN. Este paso es exclusivo de estos modelos de CS

```
card <- scorecard(bins_train, m1)
```

variable	bin	count	count_distr	neg	pos	posprob	woe	bin_iv	total_iv	breaks	is_special_values	points
Time	[~Inf,30000)	11395	0.06682736	11346	49	0.0043001316	0.923304704	9.339604e-02	0.3027489	30000	FALSE	22
Time	[30000,40000)	12553	0.07361859	12543	10	0.0007966223	-0.766227983	3.021969e-02	0.3027489	40000	FALSE	-18
Time	[40000,85000)	61728	0.36201133	61623	105	0.0017010109	-0.006725162	1.631823e-05	0.3027489	85000	FALSE	0
Time	[85000,110000)	8813	0.05168491	8772	41	0.0046522183	1.002356901	8.908773e-02	0.3027489	110000	FALSE	24
Time	[110000,125000)	14296	0.08384062	14288	8	0.0005595971	-1.119628817	6.330402e-02	0.3027489	125000	FALSE	-26
Time	[125000, Inf)	61729	0.36201720	61650	79	0.0012797875	-0.291675712	2.672512e-02	0.3027489	Inf	FALSE	-7

- Se realiza un escalado de los datos a puntos que se pueden sumar:

- Control sobre el incremento las probabilidades: $b = \frac{pdo}{\ln(2)}$

- Control sobre el efecto de escala de las puntuaciones finales: $a = TargetPoints + b \cdot \ln(TargetOdds)$

Decisiones a tomar

¿Por qué no se sacan tarjetas de puntuación en los modelos de ML?

Desarrollo de la demo CS con R

- Paso 8: TARJETA DE PUNTUACIÓN. Este paso es exclusivo de estos modelos de CS
- Una vez se toman las anteriores decisiones, se deduce un nivel base de puntuaciones y el aporte por variable y de cada uno de sus tramos según las siguientes fórmulas:

$$basepoints = a - b \cdot \alpha_0$$

$$points_{v,i} = -b \cdot \alpha_v \cdot woe_{v,i}$$

Nótese que en el modo de puntuar:

- Hay un coeficiente que depende de la fórmula global
- Otro depende de la variable
- Y finalmente otro depende del tramo

Desarrollo de la demo CS con R

- Conversión Score - PD La construcción seguida permite vincular mediante una fórmula de tipo exponencial, el valor del scoring asignado a un individuo con su probabilidad de default

$$p = \frac{1}{1 + e^{\left[\frac{1}{b}(score - a)\right]}}$$

- El conocimiento de la anterior fórmula permite establecer la biyectividad scoring-probabilidad. En muchas ocasiones, los proveedores externos no ofrecen información sobre los valores de a y b , aunque pueden ser deducidos. Como analista debemos pedirlos.
- Nótese en la función *scorecard* cómo los valores por defecto de sus parámetros permiten deducir íntegramente los valores de a y b

<code>bins</code>	Binning information generated from <code>woebin</code> function.
<code>model</code>	A glm model object.
<code>points0</code>	Target points, default 600.
<code>odds0</code>	Target odds, default 1/19. Odds = $p/(1-p)$.
<code>pdo</code>	Points to Double the Odds, default 50.
<code>basepoints_eq0</code>	Logical, Defaults to FALSE. If it is TRUE, the basepoints will equally distribute to each variable.
<code>digits</code>	The number of digits after the decimal point for points calculation. Default 0.

PARTE 3

El ML ¿Son los padres?



¿Qué es lo que nadie hace en Machine Learning?

- Es muy habitual ver en linkedin un montón de post, de algunos/as flipados/as dando definiciones absurdas sobre diferencias entre modelos de estadística tradicional, de Minería de Datos, de Machine Learning ... Y por supuesto ahora, de Inteligencia Artificial
- El manejo de los términos es importante, porque después cae en manos de políticos, periodistas y demás, llega a la gente de la calle y se lía
- Cuando se pasó de la era de la *Minería de Datos* antes del 2010 aprox. A la del ML, pensaba yo en el mundo de los modelos ... **¿Será que a partir de ahora los modelos van a aprender dinámicamente conforme llegue nueva información? ¿Se re-entrenarán automáticamente? ...**
- En este caso debo admitir que no he encontrado muchos sitios donde se haga lo anterior, y los resultados tampoco han sido espectaculares.
- Además, en el caso de banca – seguro y en entornos regulados, básicamente el paradigma no ha llegado con todas sus consecuencias sólo ha sido simple cambio

Algoritmos de analítica avanzada

- Por tanto ... **¿Qué ha ocurrido en la realidad?**
- Pues lo que ha sucedido es que desde principios del 2000 y sobre todo a partir del 2010 – 2012 han surgido una pléyade de nuevos algoritmos avanzados como: Regresiones Ridge / Lasso, SVM, Random Forest, Xgboost, ... pero **tras ser entrenados, la máquina ya no aprende, en todo caso puntúa, estima, ordena observaciones, ... pero en base a unos datos ya dados en un momento del pasado.** Esto no es lo que me habían prometido con el ML ...
- Lo que se va a desarrollar aquí son 2 líneas muy enfocadas a los modelos CS vistos anteriormente:
 - ¿Qué algoritmos pueden completar a la metodología CS en cuanto a más opcionalidad para el DS y cómo conseguirlo con R?
 - ¿Cómo se implementaría realmente un marco ML y cómo debería ser éste en la realidad?

¿Cómo usar algoritmos avanzados bajo un CS?

- La interpretabilidad y versatilidad de un modelo de CS que lo hace distintos a los demás, es que al final se genera lo que se conoce como *tarjeta de puntuación*, esta tarjeta convierte una fórmula exponencial, la de la regresión logística subyacente, en una suma de puntuaciones directa

$$P[Y = 1] = \frac{1}{1 + e^{\alpha_0 + \alpha_1 WoE_{Age}}}$$



Variable	Score	Raw score	WOE	Score	Ratio	O/R	OCR
Pais	100	18.40%	-0.019	20			
	100-100	17.80%	-0.083	40			
	100-1000	11.2%	0.245	40	-0.4032	0.2775	10.90%
	1000-1700	11.10%	0.304	51			
	1700+	7.1%	0.061	17			
Trat	400	22.74%	-0.163	20			
	400-470	16.20%	0.134	40			
	470-700	9.11%	0.734	59	-0.0724	0.2607	35.91%
	700+	6.8%	1.480	74			
	1	34.18%	-0.021	16			
Ratio_AC_FMS	2	22.44%	-0.080	40			
	3	16.89%	0.017	40	-0.0710	0.1871	5.16%
	4	12.20%	0.017	40			
	5	27.48%	-0.005	31			
	6	22.37%	-0.032	20			
Nacionalidad	1	18.70%	-0.100	40			
	4	16.92%	0.040	40	-0.7051	0.1246	16.86%
	5	12.20%	0.003	32			
	6	7.80%	0.007	38			
	7	27.44%	-0.040	38			
Incidencias	CON INCIDENCIAS	14.51%	0.196	40	-0.0177	0.1107	6.56%
	SIN INCIDENCIAS	16.6%	-0.048	42			
	10	18.84%	-0.138	40			
	20-29	16.84%	-0.138	40	-0.0715	0.0802	5.92%
	30-39	14.21%	0.022	50			
Antigüedad	0-9	9.62%	0.086	39			
	10-19	18.40%	-0.083	40			
	20-29	11.2%	0.245	40			
	30-39	11.10%	0.304	51			
	40-49	7.1%	0.061	17			

- Lo anterior se consigue porque se aplican logaritmos y debido a que es la operación inversa la exponencial, se obtiene de modo directo la transformación
- Por tanto, dicho esto **¿Cuáles de entre los algoritmos avanzados que provee el ML son susceptibles de ser usados como modelo subyacente en un CS?**

Regresiones Ridge – Lasso un breve recordatorio

- Aunque los anteriores algoritmos puedan a más de uno parecerles recientes, lo cierto es que la regresión Ridge fue creada en 1.970 por Hoerl y Kennard, mientras que la Lasso resulta ser mucho más reciente y viene del año 1.996 de la mano de Robert Tibshirani co-autor del famoso libro *Introduction to Statistical Learning* donde por el año 2013 puso de moda estos modelos con un estilo accesible para un gran espectro de estudiantes y profesionales
- En estos 2 tipos de regresiones, la estimación de sus coeficientes responde meramente a la minimización de una función de costes. Por tanto, ya no se está ante el clásico algoritmo de la regresión logística que se basaba en la maximización de la función de máxima verosimilitud, sino que se introduce una función de coste a la que se le suele dotar de ciertos términos que permita incorporar elementos de aprendizaje

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))] + \lambda \sum_{j=1}^n |\theta_j|$$

Factor lambda de penalización de los coeficientes

Término de regularización, en este caso tipo l1

Regresiones Ridge vs Lasso

- En *R* el uso de estas regresiones se hace a través de la librería *glmnet* y su aplicación se hace vía 2 parámetros, el denominado *alpha* que permite elegir entre una regresión lasso, una ridge o una combinación de ambas, y el *lambda* que impone la penalización a los coeficientes y que conviene ser estimado por validación cruzada, una vez fijado el *Alpha*

`alpha` The elasticnet mixing parameter, with $0 \leq \alpha \leq 1$. The penalty is defined as

$$(1 - \alpha)/2 \|\beta\|_2^2 + \alpha \|\beta\|_1.$$

`alpha=1` is the lasso penalty, and `alpha=0` the ridge penalty.

- ¿Cuál es la mejor opción? Pues para eso hay que estudiar cada problema y previamente conocer bien en qué consiste estas alternativas y su intermedias, en 10 min ... poco más os voy a decir, aunque siempre estar el método de prueba y error

Retomando el espíritu del ML

- Con las particularidades de cada una de estas 2 técnicas y como veremos a continuación. Su uso permitiría generar sistemas dinámicos que se fueran re-actualizando por ejemplo cada 6 – 12 meses
- Ambas técnicas pueden regular el número de variables de entrada de modo automático en función de la información que va entrando y por tanto, permite actualizar el algoritmo conforme entran nuevos datos:
 - Hacer que determinadas variables cobren más importancia frente a otras en función de los datos
 - En el caso de la Lasso puede incluso seleccionar y anular de modo automático determinadas variables
 - Si el elenco de variables es suficientemente amplio, el algoritmo es rápido de entrenar y permitiría que la arquitectura del modelo estuviese actualizado y adaptado a la información más reciente

Obteniendo la scorecard

- Sin embargo, faltan 2 detalles:
 - El primero es sencillo y consiste en hacer uso de un *binning* y de la *transformada woe*
 - El segundo es algo más complejo:

¿Cómo obtener la scorecard a partir de los parámetros dados por la librería anteriormente citada?

- La actual librería *scorecard* no dispone, desgraciadamente de una función para tal fin, sin embargo, si váis a este enlace podéis obtener una función que yo he mismo he diseñado y con una salida totalmente equivalente. Os la podéis descargar de:

https://github.com/FJROAR/R-function-scorecard_glmnet

La función `scorecard_glmnet`

- Es una función que actualmente está en una fase 0 aunque dependiente de la librería *scorecard*, se verá en la demo cómo se usa:

```
scorecard_glmnet <- function(bins,
                             model,
                             points0 = 600,
                             odds0 = 1/19,
                             pdo = 50,
                             lambda = 0){

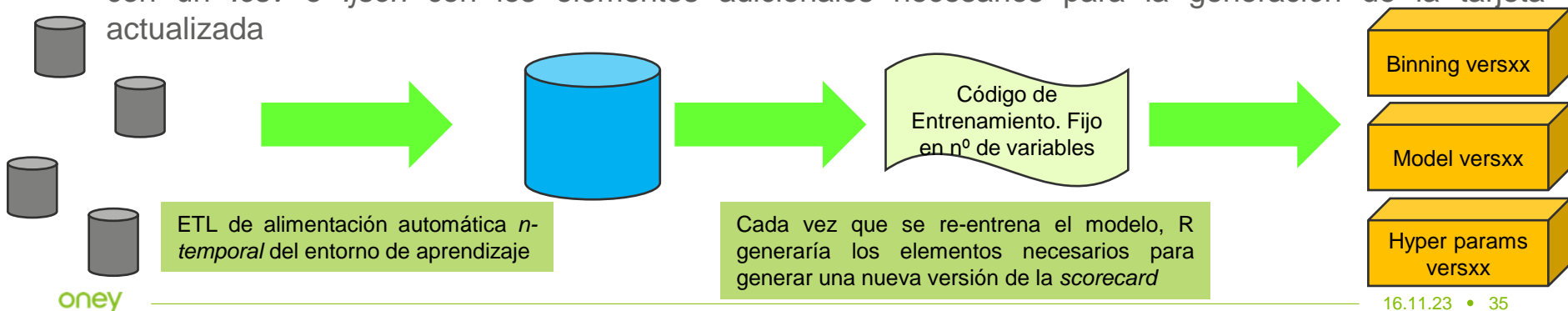
  #Inputs:
  #bins: Binning of the scorecard model
  #mod: glmnet model
  #lambda: Optimal lambda associated. The value is not needed

  #Outputs: A list with all the information required by scorecard_ply
```

- Obsérvese que análogamente a la función *scorecard()* admite los parámetros de calibrado de tarjeta de puntuación *points0*, *odds0* pero además hace uso del coeficiente *lambda* que es el hiperparámetro que permite controlar, en función de la nueva información que se considere, la penalización o la re-estimación de los coeficientes de las variables que se considere

Proceso de ML

- Se considera una opción sencilla, dejando el campo abierto a otras más completa. Esta opción consiste en dar un primer paso que consistiría en tomar una cantidad elevada de posibles variables relacionadas con la *target* a predecir. No conviene meter “variables basura”
- Posteriormente un código R con re-entrenamiento del proceso debe estar conectado a un entorno que cada período establecido (n – meses) se retro-alimente con nueva información (conteniendo la *target*)
- El código se ejecuta y generará en cada caso una nueva versión de *binning*, *modelo serializado* junto con un *.csv* o *.json* con los elementos adicionales necesarios para la generación de la tarjeta actualizada



PARTE 4

DEMO: Scorecards bajo el paradigma ML

Librerías y preparación de datos

- Este dataset es un dataset privado, por lo que no se ofrecerá, pero si se servirá el código sin sus correspondientes variables
- En este caso se usa la librería *glmnet* para hacer uso de las regresiones tipo lasso - ridge

```
library(data.table)
library(dplyr)
library(scorecard)
library(lubridate)
library(scorecardModelUtils)
library(ggcorrplot)
library(glmnet)
```

```
df_contrglob <- fread("../data/df_contrglob.csv") %>%
  filter(CTRA_INSTMNUM >= 10)
```

```
list_var = c("CTRA_ID",
             "CTRA_INSTMNUM",
             "CTRA_HH2",
```

```
#Feature Engineering
```

```
df_contrglob$CUST_DOMAIN2 <- ifelse(is.na(df_contrglob$CUST_DOMAIN) == T,
df_contrglob$CUST_DOMAIN2 <- ifelse(df_contrglob$CUST_DOMAIN %in%
  c("yahoo", "hotmail", "msn", "decatl
df_contrglob$CUST_DOMAIN2 <- ifelse(df_contrglob$CUST_DOMAIN %in%
  c("telefonica", "gmai", "gmaio", "gma
```

Selección de la información

- Como se va a dar un paso al ML se supone una restricción temporal de la información

```
df_mod_tot <- df_contrglob %>%  
  filter(substr(CTRA_CREAHH2, 1, 7) <= "2022-05") %>%  
  select(all_of(list_var))
```

Entrenamiento del modelo I

- Selección training – test, en un entorno ML, convendría ir a unos ratios 90% - 10%

```
df_list_tot <- split_df(df_mod_tot, y = "TARGETefin", ratios = c(0.7, 0.3), seed = 30)
label_list_tot <- lapply(df_list_tot, function(x) x$TARGETefin)

df_train_tot <- df_list_tot$train
df_test_tot <- df_list_tot$test

df_list_tot$train <- df_train_tot %>% select (-CTRA_ID)
df_list_tot$test <- df_test_tot %>% select (-CTRA_ID)
```

Entrenamiento del modelo II

- Binning y su serialización

```
#Elemento a serializar
bins_train <- woebin(df_list_tot$train, y = "TARGETefin",
                     count_distr_limit = 0.01)

save(bins_train, file = "Rmodel/bins_vers01.RData")
```

- Transformación WoE: reducción de valores en variables continuas y conversión de variables discretas a numéricas

```
df_woe_list = lapply(df_list_tot, function(x) woebin_ply(x, bins_train))
```

Entrenamiento Modelo Lasso

- Entrenamiento de un modelo tipo Lasso

```
input1 <- as.matrix(df_woe_list$train[,-"TARGETefin"])
target1 <- df_woe_list$train$TARGETefin
```

```
alpha = 1 #Regression lasso (= 1) o ridge (= 0)
```

```
cv_m2 <- cv.glmnet(input1, target1, family = "binomial", alpha = alpha)
lambda_optimo <- cv_m2$lambda.min
```

```
m2 <- glmnet(input1, target1, family = "binomial", alpha = alpha, lambda = lambda_optimo)
```

```
coeficientes <- coef(m2, s = lambda_optimo)
```

	s1
	-2.6580726
M_woe	0.3459431
	1.1991383
AR_woe	1.3161982
_woe	0.5609196
NICNUM_ITEM_woe	0.4503850
_ITEM_woe	0.3373376
_ITEM_woe	0.5260066
ITEM_woe	.
SE_AMT_woe	0.6349215
	0.8586961
	0.9244687
woe	0.7302124
ER_woe	0.9217916
woe	0.6691762
P_woe	.
IRATION2_woe	0.5614528
E_woe	0.6699359

```
save(m2, file = "Rmodel/modlasso_vers01.RData")
```

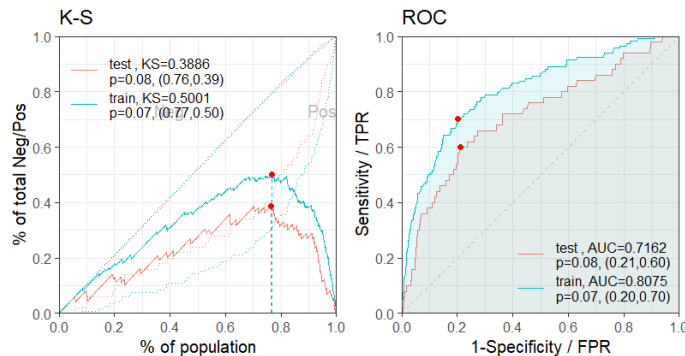
```
write.table(lambda_optimo, "Rmodel/modlasso_hyper_vers01.csv", row.names = F)
```


Entrenamiento Modelo Lasso

- Entrenamiento de un modelo tipo Lasso: Predicción

```
df_woe_list_2 <- df_woe_list
df_woe_list_2$train <- as.matrix(df_woe_list$train %>% select(-TARGETefin))
df_woe_list_2$test <- as.matrix(df_woe_list$test %>% select(-TARGETefin))

pred_list2 = lapply(df_woe_list_2, function(x) predict(m2, newx = x, type='response'))
perf = perf_eva(pred = pred_list2, label = label_list_tot, show_plot = c("ks", "roc"),
  pred_desc = F)
```



- Llamada a la función de scorecard para modelos de regresión avanzada

```
source("functions/scorecard_glmnet.R")
```

Entrenamiento Modelo Lasso

▪ Tarjeta de puntuación final

Name	Type	Value
card2	list [18]	List of length 18
basepoints	list [1 x 4] (S3: data.table, data.frame)	A data.table with 1 row and 4 columns
CTRA_INSTLMNUM	list [2 x 13] (S3: data.table, data.frame)	A data.table with 2 rows and 13 columns
CTRA_HH2	list [3 x 13] (S3: data.table, data.frame)	A data.table with 3 rows and 13 columns

CTRA_HH2	list [3 x 13] (S3: data.table, data.frame)	A data.table with 3 rows and 13 columns
variable	character [3]	'CTRA_HH2' 'CTRA_HH2' 'CTRA_HH2'
bin	character [3]	'Gr1' 'Gr2' 'Gr3'
count	integer [3]	54 1804 60
count_distr	double [3]	0.0282 0.9406 0.0313
neg	integer [3]	52 1697 51
pos	integer [3]	2 107 9
posprob	double [3]	0.0370 0.0593 0.1500
woe	double [3]	-0.5332 -0.0389 0.9903
bin_iv	double [3]	0.00637 0.00140 0.04747
total_iv	double [3]	0.0552 0.0552 0.0552
breaks	character [3]	'Gr1' 'Gr2' 'Gr3'
is_special_values	logical [3]	FALSE FALSE FALSE
points	double [3]	46.12 3.37 -85.66

Proceso de ML I

- Y llega información de un nuevo mes !!!!!

```
df_mod_tot <- df_constrglob %>%
  filter(substr(CTRA_CREAHH2, 1, 7) <= "2022-06") %>%
  select(all_of(list_var))
```

```
save(bins_train, file = "Rmodel/bins_vers02.RData")
```

```

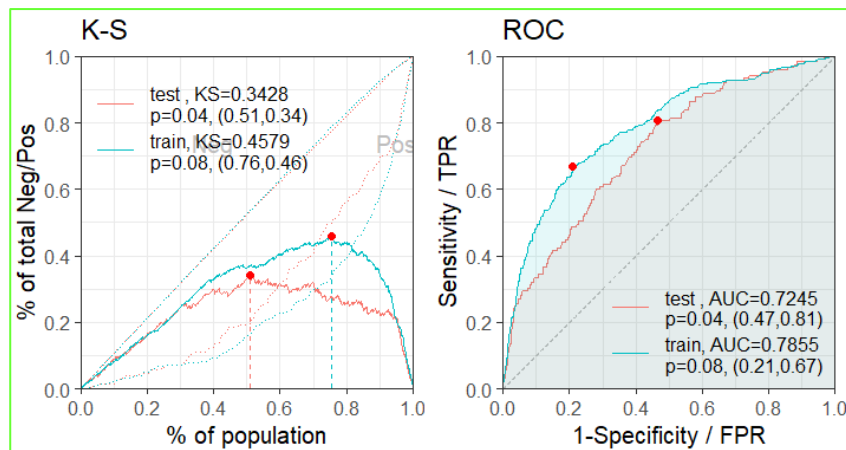
s1
-2.5662146
_woe      0.4277185
          1.2054148
_woe      0.5961130
_woe      0.8593836
CNUM_ITEM_woe 0.5619447
ITEM_woe     0.7279242
ITEM_woe     0.4089259
ITEM_woe     0.5076442
E_AMT_woe    0.8350239
           0.9293248
           0.9953928
           0.7850048
_woe      1.1231331
           0.4928423
_woe     -0.5477563
ACTION2_woe  0.6418873
_woe      0.5159492

```

```
save(m2, file = "Rmodel/modlasso_vers02.RData")
write.table(lambda_optimo, "Rmodel/modlasso_hyper_vers02.csv", row.names = F)
```

Proceso de ML II

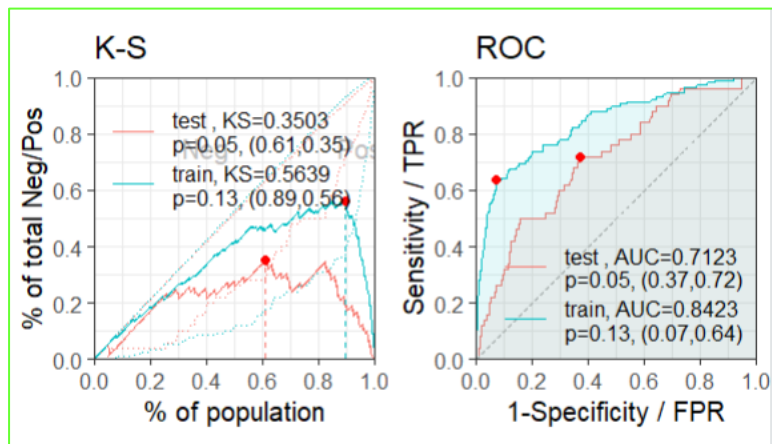
- Y llega información de un nuevo mes !!!!!



CTRA_HH2	list [3 x 13] (S3: data.table, data.frame) A data.table with 3 rows and 13 columns		
variable	character [3]	'CTRA_HH2' 'CTRA_HH2' 'CTRA_HH2'	
bin	character [3]	'Gr1' 'Gr2' 'Gr3'	
count	integer [3]	106 3957 137	
count_distr	double [3]	0.0252 0.9421 0.0326	
neg	integer [3]	102 3686 116	
pos	integer [3]	4 271 21	
posprob	double [3]	0.0377 0.0685 0.1533	
woe	double [3]	-0.6593 -0.0308 0.8703	
bin_iv	double [3]	0.008316 0.000881 0.035886	
total_iv	double [3]	0.0451 0.0451 0.0451	
breaks	character [3]	'Gr1' 'Gr2' 'Gr3'	
is_special_values	logical [3]	FALSE FALSE FALSE	
points	double [3]	57.33 2.68 -75.68	

ANEXO: ¿Y un random forest?

- Con la misma información no se consiguen mejoras significativas



PARTE 5

Conclusiones Finales



CONCLUSIONES FINALES

- La librería *scorecard* permite a día de hoy completar el ciclo de modelización para CS ahorrando mucho tiempo de programación al estadístico
- Es una librería muy flexible y optimizada para cubrir con eficiencia todas las necesidades de los CSs
- Se comporta bien ante conjuntos con elevado número de registros y variables, aunque en general en estos modelos los datasets suelen ser inferiores a 100.000 registros
- Tanto los tramos como algunos parámetros adicionales para construir tarjetas en distintas escalas, son modificables de modo trivial en el código evitando muchos errores operacionales en la construcción
- Su uso en R resulta muy sencillo y permite generar rápidamente modelos que se pueden poner en producción con extrema facilidad

CONCLUSIONES FINALES

- La metodología de scorecard se extiende de modo natural a las regresiones de tipo ridge – lasso
- Se puede extender con una función ad-hoc bastante sencilla los elementos de la librería *scorecard* y hacerlos compatibles con la *glmnet*
- **Un modelo ML debería aprender de modo autónomo a partir de los datos, no debería tratarse de unas reglas más o menos avanzadas de modo fijo**
- Los modelos ML no suelen por lo general mejorar infinitamente, pero con una buena metodología de refresco de la información pueden mantenerse siempre “frescos”, aunque su calidad dependerá de la de los datos que refresquen al modelo

A photograph of a woman in a grey t-shirt and blue jeans pushing two young girls with curly hair on a skateboard. The girls are wearing a pink and white striped shirt and a green t-shirt respectively. They are all smiling and looking forward. The background is a blurred outdoor setting with trees and a fence.

**¡Muchas Gracias por su
atención!**