Construcción de Modelos de Credit Scoring con R

Mar 2023

FRANCISCO J. RODRÍGUEZ ARAGÓN

SCORE MANAGER at ONEY

Ph D in Statistic by Cordoba Unv.

Associate Professional Risk Manager Certified by APRM

Operational Risk Manager Certified by APRM

oney

YOUR MONEY YOUR WAY

Summary

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?

Los modelos más utilizados en la actualidad

- En esencia los modelos de Credit Scoring no son más que una reformulación de un modelo de regresion 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 ser 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*
- Además se va a exigir un alto grado de interpretación y de estabilidad:
  - Estabilidad a nivel de tramos de puntuaciones de score totals
  - 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	13,23%	0,305	49			
	2000-3700	11,10%	0,504	51			
	3700+	7,12%	0,992	57			
Triad	-650	22,74%	-0,353	39	-0,6724	0,2667	35,91%
	650-675	15,29%	0,136	48			
	675-700	9,11%	0,724	59			
	700+	4,60%	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%	0,327	49			
Nacionalidad	1	27,48%	-0,605	33	-0,7051	0,1246	18,56%
	2	22,37%	-0,332	39			
	3	18,75%	-0,110	43			
	4	16,50%	0,046	46			
	5	13,25%	0,303	50			
	6	7,50%	0,937	58			
Incidencias	CON INCIDENCIAS	27,40%	-0,601	33	-0,5877	0,1119	6,56%
	SIN INCIDENCIAS	14,53%	0,196	48			
Antiguedad	-11	19,61%	-0,165	42	-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*. Las variables son tomadas en general sobre un individuo, empresa, transacción, etc y por tanto la suma de los valores de todas las variables consideradas sobre dicho elemento de la población es lo que se denomina, *scoring* de dicho elemento

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

Ejemplo de score básico formado por una única variable (o un único score parcial) suponiendo datos limpios y target adecuadamente construida:

Paso 1: Binning de Variables

From 18 to 35 years with a high rate of default
From 36 to 45 years with a low rate of default
From 45 to end with a medium rate of default

Paso 2: Transformación WoE

$$WoE = \left[ln\left(\frac{Relative\ Frequency\ of\ Goods}{Relative\ Frequency\ of\ Bads}\right)\right]$$

Paso 3: Regresión Logística subyacente

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

Paso 4: Transformación a puntos score

$$Score = Offset + Factor * Ln(odds)$$

$$odds = \frac{P(Y=0)}{P(Y=1)} = e^{\alpha_0+\alpha_1 WoE\_Edad} \qquad Factor = \frac{20}{Ln(2)}$$

Paso 5: Tarjeta de puntuación final

$$Age \in [18;35] \quad Points = 600 + 28.85 \cdot (3.45 + 5.45 \cdot -0.29) = 654.30$$
$$Age \in [36;45] \quad Points = 600 + 28.85 \cdot (3.45 + 5.45 \cdot 0.27) = 742.29$$
$$Age \in [46;Inf] \quad Points = 600 + 28.85 \cdot (3.45 + 5.45 \cdot 0) = 699.53$$

Un modelo muy interpretable

- La construcción de este tipo de modelos requiere un trameado previo de las variables: La transformación Weight of Evidence que convierte todas las variables en tramos de puntuación continua de modo que al final se saben cosas como:

$$
\begin{array}{llll}
If & Age \in [18;35] & Points = 600 + 28.85 \cdot (3.45 + 5.45 \cdot -0.29) & = 654.30 \\
If & Age \in [36;45] & Points = 600 + 28.85 \cdot (3.45 + 5.45 \cdot 0.27) & = 742.29 \\
If & Age \in [46;Inf] & Points = 600 + 28.85 \cdot (3.45 + 5.45 \cdot 0) & = 699.53
\end{array}
$$

********** Importancia de variables Voting_Classifier **********

Variables	Logit	Random_Forest
log_providerA_score	0.376384	0.366804
provider2_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	0.005851
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

- ¿Por qué un CS frente a un Random Forest?

EBA DISCUSSION PAPER ON MACHINE LEARNING FOR IRB MODELS

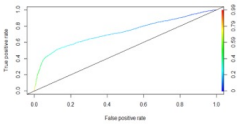
EBA/DP/2021/04

11 NOVEMBER 2021

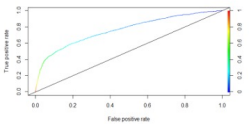
En este caso particular se observa como un Random Forest mejora la predictividad desde un 0.7238 de una regresión logística hasta un 0.7633, es decir, algo más de un 5.45%

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

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

Pasos esenciales en la construcción de un modelo de CS

¿Cómo se estima la probabilidad de Impago?

- Como en todo modelo, el paso más difícil es la construcción de una tabla que contengan *variables explicativas* junto con una *variable objetivo* o *target*

- Se trata de predecir la probabilidad de impago condicionada al valor de unas *variables explicativas*

$$\mathbf{P}(Y = 1 \mid v_1; v_2; \ldots; v_n) = ?$$

- Mientras que, en una regresión logística, el anterior número para un determinado cliente se estima del siguiente modo:

$$P(Y = 1|v_1; \ldots; v_n) = \frac{e^{a_0+a_1v_1+\cdots++a_nv_n}}{1 + e^{a_0+a_1v_1+\cdots++a_nv_n}}$$

- En un Credit Scoring, dicho valor se estima del siguiente modo (bajo las condiciones de construcción que se verán en la demo, ambas formulaciones son equivalentes):

$$Score = a + b\left[\frac{P}{1-P}\right] = a + b[a_0 + a_1v_1 + \cdots + +a_nv_n]$$

Es habitual tomar el valor de *a* igual a: 487.123 y el de *b* igual a: 28.8539. Aunque pueden tomarse otros como en el ejemplo siguiente (600; 28.85)

CS y Machine Learning

- En la construcción de un CS hay 2 elementos claves típicos de estos modelos
  - Uno es el binning de variables y su transformación a variables WoE que puede ser utilizado como transformación plausible en cualquier modelo de ML aunque es de esperar, que en aquellos ensembles basados en árboles, internamente ya se haga el citado binning
  - El otro es la transformación a puntos score, en este caso se necesita disponer de un modelo de regresión logística para que el logaritmo, se neutralice con la exponencial y entonces quede la suma aritmética simple de scoring parciales
- Teniendo en cuenta lo anterior, es factible considerar el segundo punto y añadir la familia de los modelos de regresión tipo lasso – ridge como candidatos para constituir un CS, sobre todo en los casos en los que se disponga de un número elevado de variables o en los que se quiera introducir elementos de re-entrenamiento automático con sus ventajes y posibles peligros

PARTE 3

DEMO DE LA LIBRERÍA *scorecard*

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 pasoe ya que los datos estaban 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

```
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

- Paso 5: Construcción de un modelo de regresión logística subyacente con las variables trameadas (training del modelo)

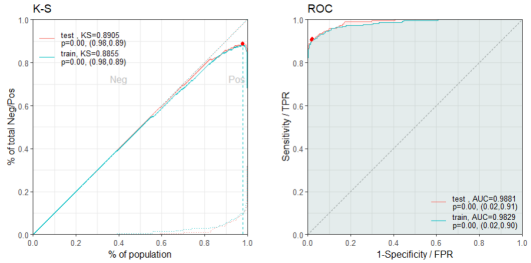
```
dt_woe_list = lapply(df_list, function(x) woebin_ply(x, bins_train))
m1 = glm( Class ~ ., family = binomial(), data = dt_woe_list$train)

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 ***
```

Desarrollo de la demo CS con R

- Paso 6: 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)
```



- Paso 7: TARJETA DE PUNTUACIÓN. Este paso es exclusivo de estos modelos de CS

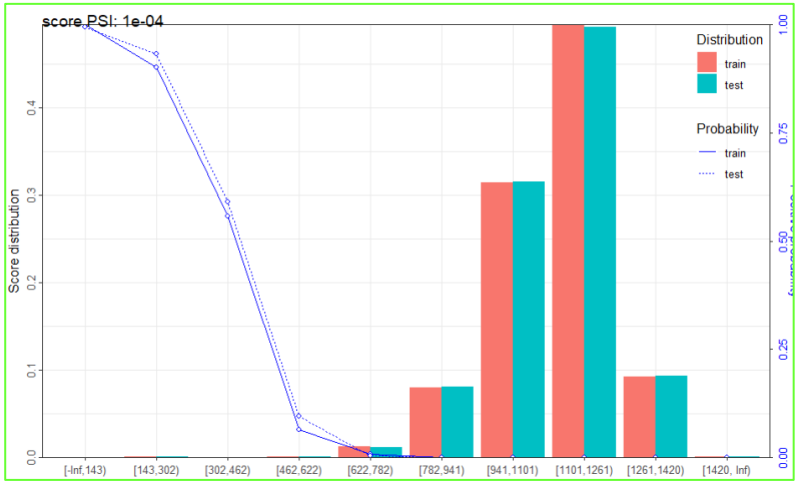
```
card <- scorecard(bins_train, m1)
```

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

Desarrollo de la demo CS con R

- Paso 8: Aplicación de la tarjeta a conjunto de datos y evaluaciones adicionales como la estabilidad poblacional training vs test

```
# Obtain Credit Scores
score_train <- scorecard_ply(df_train, card)
score_test <- scorecard_ply(df_test, card)

score_list = lapply(df_list, function(x) scorecard_ply(x, card))

# Analyze the PSI
perf_psi(score = score_list, label = label_list)
```



PARTE 4

Combinaciones de modelos

Objetivos de la combinación de modelos

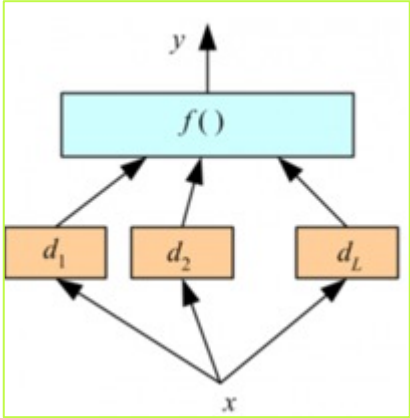
- La combinación de modelos pretende mejorar la predictividad de los modelos que intervienen en dicha “combinación”, La combinación parte de la idea base de compensar errores entre modelos y generar predicciones más robustas

- Hay distintos tipos de combinaciones que pueden ir desde los ensembles que se hacen habitualmente en ML con infinidad de modelos hasta los de tipo stacking y mixtura de modelos que suelen hacer uso de un número más reducido de modelos e incluso entre sus elementos tiene sentido considerar un ensemble

- En las siguientes slides se tratarán las ventajas y desventajas de algunas de las modalidades del segundo grupo de las anteriormente mencionadas combinaciones de modelos

Stacking de modelos

- Hay varios tipos de stacking de modelos, el más habitual es la combinación de distintos algoritmos que son utilizados como variables explicativas bajo otro algoritmo superior



(1) Este caso admite múltiples variantes que van desde cambiar los datos de origen, hasta las variables a utilizar

(2) En esencia se crea una capa intermedia de modelos que reciben los inputs y cuyos outputs serán las variables de entrada del modelo superior, con parámetros que tratan de predecir a similares o distintas variables objetivo respecto a los modelos base

Mixturas y quasi-stacking

- En las mixturas puras de modelos. Los outputs de los modelos que intervienen se aplican bajo una combinación lineal con parámetros (que por lo general suman 1) de los modelos que intervienen en la combinación. Por lo general es una media ponderada, pero puede haber otras opciones sencillas

Votación por mayoría

Votación por pluralidad

Votación ponderada

- Sin embargo, resulta ser más habitual de lo que parece que en el mundo de la empresa se confundan modelos por variables y en consecuencia crean quasi-stacking donde se hace uso de variables explicativas, junto con un (o varios) modelo(s), en general vendido por un tercero que se trata también como una variable explicativa, lo que resulta tan complejo y problemático como el caso del stacking

$$Modelo = F(v_1; \dots; v_n; Modelo\ Adicional_1; Modelo\ Adicional_m)$$

PARTE 5

DEMO DE COMBINACIONES DE MODELOS

Ejemplo con R: Mixtura frente quimeras

- En el siguiente código se desarrolla un ejemplo donde se observa que un sistema de stacking no tiene porqué mejorar a una mixtura de modelos
- Además de lo anterior se comentará sobre los problemas de las puestas en producción tanto de los stacking como de las combinaciones de modelos que impliquen estimación de parámetros a partir de los datos
- Para todo esto se puede consultar la entrada del blog, donde el ejemplo se hizo en Python, aunque aquí se hace algo similar bajo entorno R

POC: Separación de variables en 2 data frames

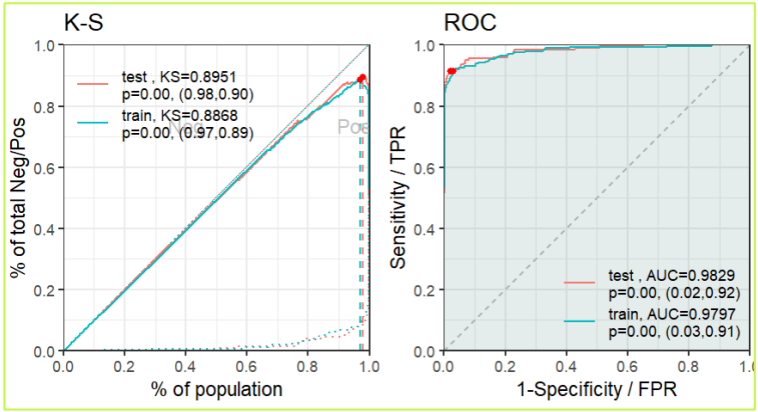
- En general no se va a tener un mismo conjunto de datos para hacer esto, aquí se tiene la suerte de que sí. Además, en los stacking – mixturas reales, el modelo proporcionado (vendido) por un tercero, tendrá targets distintas, por lo que cabe esperar peor comportamiento que en este caso

```
df_train1 <- df_train[,c("Time", "V1", "V2", "V3", "V4", "V5", "V6", "V7",
                         "V8", "V9", "V10", "V11", "V12", "V13", "V14", "V15",
                         "V16", "V17", "V18", "V19", "V20", "Class" )]
df_test1 <- df_test[,c("Time", "V1", "V2", "V3", "V4", "V5", "V6", "V7",
                       "V8", "V9", "V10", "V11", "V12", "V13", "V14", "V15",
                       "V16", "V17", "V18", "V19", "V20", "Class" )]

df_train2 <- df_train[,c("V21", "V22", "V23", "V24",
                         "V25", "V26", "V27", "V28", "Amount", "Class" )]
df_test2 <- df_test[,c("V21", "V22", "V23", "V24",
                       "V25", "V26", "V27", "V28", "Amount", "Class" )]
```

POC: Construcción del modelo base cliente

- Se supone que el cliente tiene un único modelo que desea mejorar a partir de “algo maravilloso” que le vende un proveedor externo

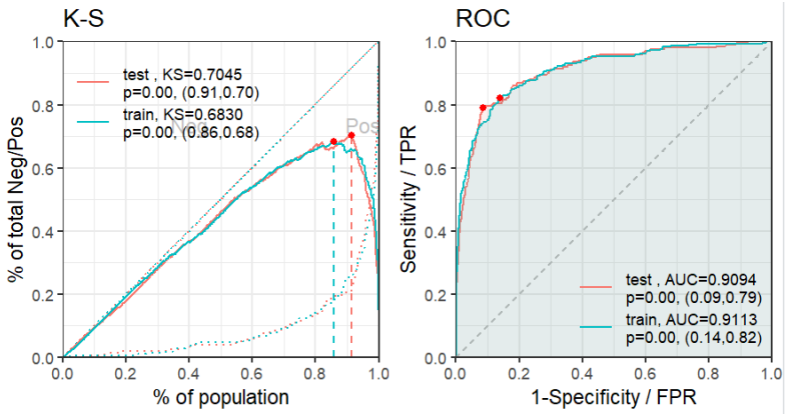
```
#WoE en train

df_list1 = list(df_train1, df_test1)

bins_train1 <- woebin(df_train1, y = "Class")
dt_woe_list1 = lapply(df_list1, function(x) woebin_ply(x, bins_train1))
names(dt_woe_list1[[1]])

m1 = glm( Class ~ ., family = binomial(), data = dt_woe_list1[[1]])

summary(m1)
```



POC: Modelo proveedor

- Aquí viene el otro modelo que tiene el siguiente comportamiento:

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.39861    0.08648 -73.993  < 2e-16 ***
V21_woe      0.74522    0.04581  16.266  < 2e-16 ***
V22_woe     -0.55889    0.24496  -2.282   0.0225 *
V23_woe      0.02892    0.07649   0.378   0.7053
V24_woe      0.95461    0.11829   8.070 7.03e-16 ***
V25_woe      0.02434    0.14378   0.169   0.8655
V26_woe      0.83649    0.16451   5.085 3.68e-07 ***
V27_woe      0.69617    0.04582  15.194  < 2e-16 ***
V28_woe      0.29251    0.05949   4.917 8.79e-07 ***
Amount_woe   0.89947    0.09598   9.371  < 2e-16 ***
```



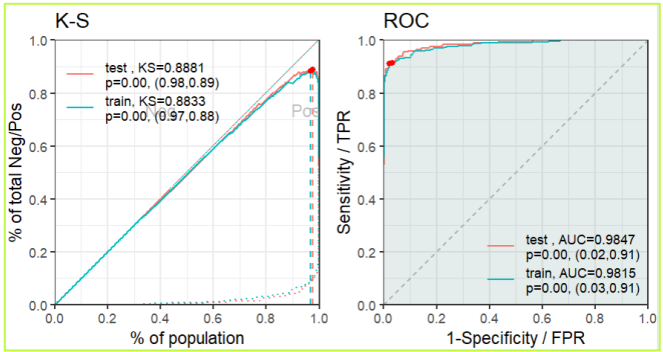
¿Cómo se haría la integración? Nótese que este segundo modelo tiene un comportamiento peor, pero aún así puede ser útil

POC: Cuasi - stacking

- Esta opción suele ser la más habitual en muchas empresas. En ocasiones se hace así aconsejados por el proveedor del modelo, el cuál de este modo asegura una dependencia duradera

- En esta integración, lo que se hace es tratar el modelo de proveedor, como una variable más, por tanto interesa las previsiones del modelo que entran a formar parte de las variables explicativas

```
df_train3 <- df_train1
df_test3 <- df_test1

df_train3$stacking <- pred_list2[[1]]
df_test3$stacking <- pred_list2[[2]]
```



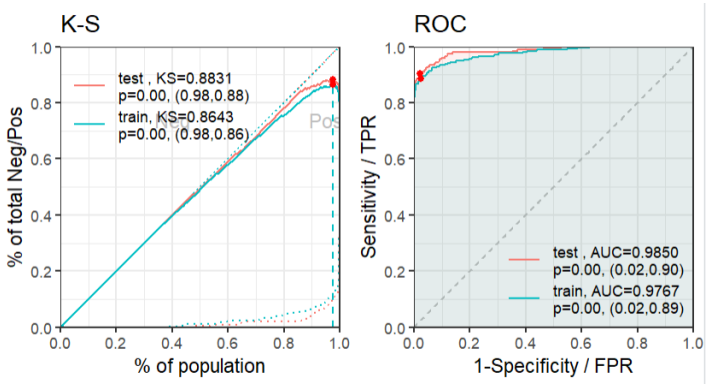
Aquí se ha mejorado ligeramente (es difícil con un 98% obtener mejoras) aunque en un ejercicio en mi blog con otro dataset, se obtuvo incluso empeoramiento : https://fjroar.wixsite.com/cosasveredes/post/el-peligro-del-stacking-de-modelos-en-riesgo-de-cr%C3%A9dito

POC: Mixtura

- Es una opción mucho más simple que la anterior, y en general puede dar resultados similares e incluso mejores

- Se tienen por separados los modelos y es fácil cambiar de proveedor o incluso no considerarlo si los resultados no son óptimos o empeoran en cualquier momento

```
#Mixtura de modelos del m1 con el m2 en este caso no hay que re-estimar
#Se elije el nivel de mixtura y los modelos conservan sus características

pred_mixt <- list((0.8*pred_list1[[1]] + 0.2*pred_list2[[1]]),
                  (0.8*pred_list1[[2]] + 0.2*pred_list2[[2]]))
names(pred_mixt)[1] = "train"
names(pred_mixt)[2] = "test"
```

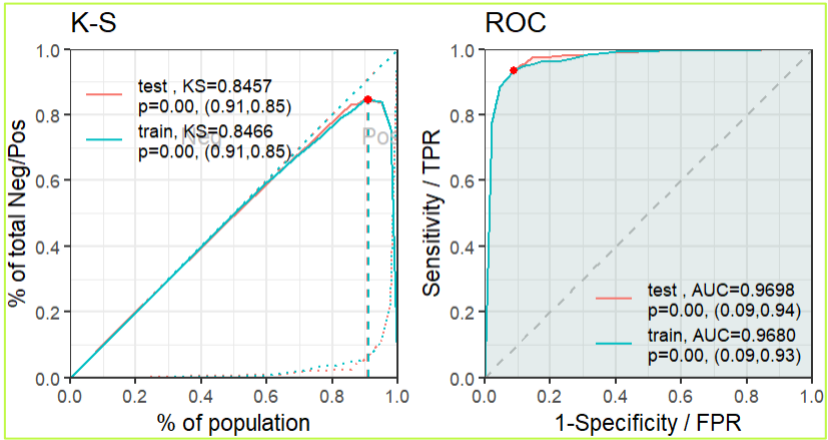


Esta opción debería ser considerada como best-practice y preferible siempre salvo que la mejora de un stacking sea manifiestamente superior. Recuerde que un stacking es más complejo ponerlo en producción

POC: Stacking completo variante 1

- Esta sería la solución que habitualmente se vende desde el mundo “DS” como una opción poderosa, no lo tengo tan claro y se suele usar poco (por suerte)

```
#Se toma la target de alguno de los modelos
df_stacking_train <- df_train1[,c("Class")]
df_stacking_test <- df_test1[,c("Class")]

#Se toma la predicción del modelo 1 como input

df_stacking_train$stacking1 <- pred_list1[[1]]
df_stacking_test$stacking1 <- pred_list1[[2]]

#Lo mismo con el modelo 2

df_stacking_train$stacking2 <- pred_list2[[1]]
df_stacking_test$stacking2 <- pred_list2[[2]]
```

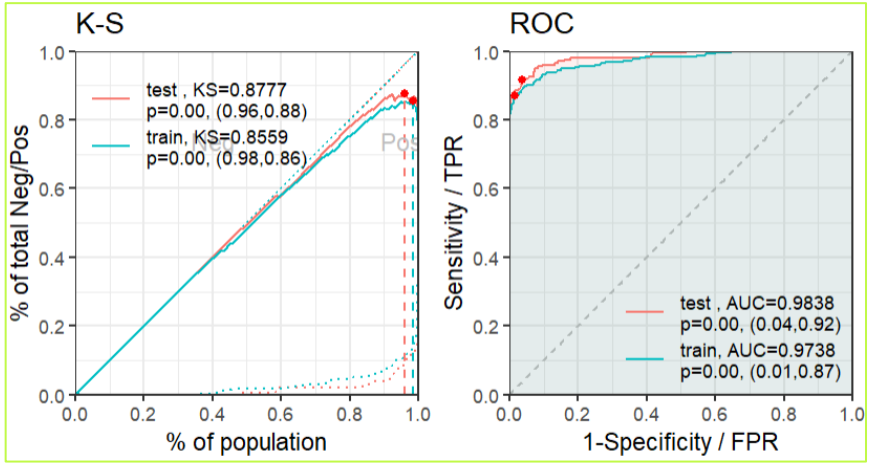


El resultado es manifiestamente inferior. En muchas oportunidades que he tenido a nivel académico y empresarial me suele pasar esto, pero se insiste en sus bondades en la literatura actutal

POC: Stacking completo variante 2

- En este caso, la construcción del modelo no hace binning ni transformación de variables explicativas y sólo se introducen como explicativas sin transformación ninguna (sería el stacking puro)

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -7.7838     0.1175  -66.22   <2e-16 ***
stacking1    11.8188     0.3682   32.09   <2e-16 ***
stacking2     6.1818     3.1535    1.96     0.05 *
```



Esta variante es mejor que la anterior, pero no llega al nivel de cuasi mixtura, ni de la mixtura

De verdad, ¿Hay que complicar tanto las cosas? …

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

- Cuando se usan modelos de proveedores externos debe medirse siempre los resultados y comparar opciones. No siempre la opción más complicada es la mejor
- Es recomendable integrar modelos de terceros mediante mixturas. Cualquier técnica que internalice un modelo como una variable explicativa, provoca una dependencia del proveedor clara, ya que su supresión significaría re-estimar el modelo al completo
- Aquí se ha usado un dataset libre con bastantes datos, en aplicaciones reales, las diferencias son aún más abultadas llegándose a observar stacking de modelos que funcionan incluso peor que los modelos por separado, aunque a nivel de documentación se afirme lo contrario sin reproducibilidad ninguna
- Un stacking de modelos es más complejo de poner en producción frente a una mixtura, si el stacking no mejora significativamente, conviene no aplicarlo
- Usar información externa puede ser beneficioso, pero debe medirse el coste – beneficio de ésta, ya que se por lo general, se paga por consulta que se le hace al modelo

oney
YOUR MONEY YOUR WAY