BBVA | GLOBAL RISK MANAGEMENT

Gravity Auto - SVM

Documento Metodológico

Metodologías de Riesgo, Scorings & Ratings

México, D.F., Julio 2015

Índice

Control de Versiones

Versión	Responsable	Fecha	Observaciones	Fecha
01	Campos Viana Pablo	Julio 2015	-	-

1. Introducción

1.1. Ámbito de aplicación

El ámbito de aplicación de la herramienta cuyo procedimiento de desarrollo se detalla en este documento es el siguiente

- Tipología de riesgo: Crédito
- Segmento: Minorista
- Etapa de riesgo: Admisión
- Área de negocio: México

1.2. Período de aplicación

La vigencia de la herramienta se define inicialmente desde el momento de su implantación hasta que se produzca la próxima revisión de la herramienta.

1.3. Antecedentes

El proyecto de admisión **Gravity** se desarrolló a inicios de 2015 con la finalidad de ser más efectivos en los tiempos de respuestas a los clientes, así como adquirir información de Prospector que pudiera enriquecer los modelos. Por otra parte, por restricciones de tiempo y de negocio se decidió utilizar el *Score Reactivo* como variable explicativa.

El desarrollo **Gravity 2.0** surge con la finalidad de probar un nuevo método para realizar el score de admisión, así como incluir las variables que componen el Score Reactivo en lugar de incluir éste como variable explicativa, contando con un mayor tiempo para la realización del proyecto.

Los modelos finales para los segmentos 1 y 4 fueron obtenidos a través del método de Máquinas de Soporte Vectorial (**SVM**), mientras que los modelos finales para los segmentos 5, 6, 2 fueron obtenidos a través de la metodología usual, la regresión logística con variables trameadas.

2. Máquinas de soporte vectorial

2.1. Introducción

Las máquinas de soporte vectorial (**SVM**) son un conjunto de métodos utilizados en el Aprendizaje Estadístico para modelar problemas de clasificación y de regresión. En su formulación más tradicional para problemas de clasificación, dado un conjunto de observaciones en donde cada una pertenece a una de dos clases, el método de SVM construye un modelo que asigna nuevas observaciones a alguna de las dos clases. Además de utilizarse para clasificación lineal, el método puede ser utilizado para realizar clasificación no lineal mediante el **truco del Kernel**. El método, gracias a su gran flexibilidad, es ampliamente utilizado en aplicaciones de Inteligencia Artificial, especialmente en Computer Vision, Speech Recognition y Text Categorization. Debido a su formulación y a sus propiedades, es posible construir un *score* para riesgo de crédito, además de dar una interpretación en términos de la similitud entre los clientes evaluados.

2.2. Mapeo explícito de las observaciones

La complejidad del conjunto de datos de entrenamiento afecta directamente el desempeño de cualquier método de aprendizaje que pueda ser utilizado. En ocasiones, el uso de cierta clase de métodos no es apropiado para construir una función de predicción para un conjunto de datos dado. Tal puede ser el caso de métodos lineales (como regresión logística, o análisis discriminante lineal) cuando las variables a utilizar en la modelización de cierto fenómeno no guardan una relación lineal con dicho fenómeno. Por ejemplo, utilizar ciertos ratios financieros cuya relación con la probabilidad de incumplimiento no es monótona y/o lineal, puede no ser adecuado cuando se utiliza una regresión logística. Por esta razón, realizar una transformación de las variables puede resultar de gran ayuda para la construcción de una función de predicción.

La razón más importante para transformar los datos es que el espacio al que se realiza el mapeo puede estar dotado de cierta estructura que pueda ser explotado por el método de aprendizaje a utilizar; en particular dicha estructura puede ser lineal.

De esta forma, el conjunto de datos de entrenamiento está dado por

$$\mathcal{S} = \{x_i, y_i\}_{i=1}^{n}, \quad x_i \in \mathcal{X}, y_i \in \mathcal{Y} \tag{1}$$

y su representación después de aplicar alguna función de transformación $\Phi : \mathcal{X} \to \mathcal{H}$ es

$$\mathcal{S} = \{\Phi(x_i), y_i\}_{i=1}^{n}, \quad \Phi(x_i) \in \mathcal{H}, y_i \in \mathcal{Y}. \tag{2}$$

en donde $\mathcal{X}$ es el espacio de las observaciones, $\mathcal{Y} = \{-1, 1\}$ es el conjunto de ambas clases y $\mathcal{H}$ es el espacio al que son mapeadas las observaciones.

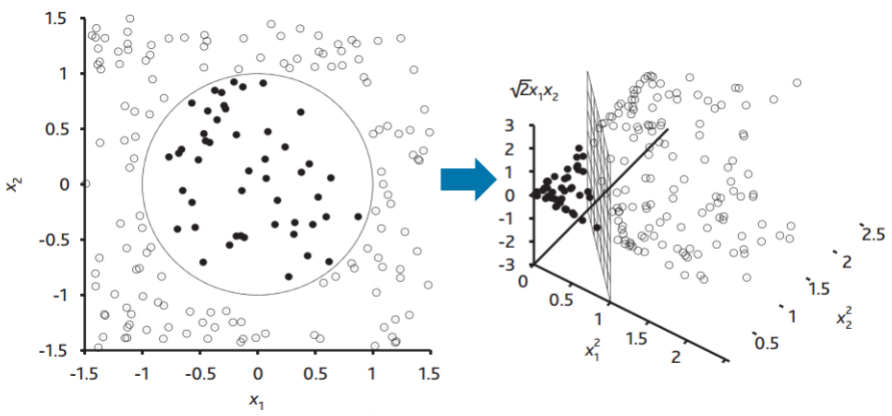

Figura 1: Mapeo explícito de las observaciones. En la figura de la izquierda, la función de decisión corresponde en R^2 a $x^2 + y^2 = 1$. En la figura derecha, los mismos datos son transformados al espacio R^3 mediante un mapeo cuadrático: $\Phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$.

2.3. Mapeo implícito a través de Kernels

Realizar el mapeo de las observaciones de forma individual tal como se ha descrito, es decir, realizar la transformación de cada x_i hacia $\Phi(x_i)$ tiene la desventaja de ser costoso computacionalmente dado que el mapeo Φ puede ser realizado hacia espacios de dimensión muy alta.

En lugar de realizar dicha transformación de forma explícita, el método SVM (y más generalmente, cualquier método basado en Kernels) representa los datos como un conjunto de evaluaciones por pares:

$$K : \mathcal{X} \times \mathcal{X} \to R \tag{3}$$

La función Kernel (o simplemente el *Kernel*) K se define sobre un espacio posiblemente de dimensión infinita. Cuando el dominio se restringe a un conjunto finito de datos (i.e. la muestra $\mathcal{S}$), entonces el Kernel finito está dado por:

$$K_{\mathcal{S}} = x_i \times x_j \to R \quad 1 \leq i, j \leq n \tag{4}$$

De esta formal, el Kernel (finito) puede ser represantado como una matriz cuadrada de dimensión $n \times n$ en donde $K_{\mathcal{S}}(x_i, x_j) = k_{i,j}$.

Asumiendo que las observaciones se encuentran definidas en un espacio $\mathcal{X}$ con producto interior, se puede construir una función de comparación lineal tomando el producto interior de modo que

$$K(x_i, x_j) = \langle x_i, x_j \rangle_{\mathcal{X}} \tag{5}$$

o bien, en espacios reales (i.e. $\mathcal{X} = R^n$):

$$K(x_i, x_j) = x_i \cdot x_j. \tag{6}$$

Debe considerarse que geométricamente, el producto punto entre dos vectores x_i y x_j se relaciona con el ángulo que existe entre ellos, dado que

$$\frac{x_i \cdot x_j}{||x_i||||x_j||} = \cos\theta \tag{7}$$

de modo que puede interpretarse al Kernel como una medida de similitud entre sus argumentos.

Por otra parte, puede probarse que bajo condiciones muy generales sobre los Kernels, este enfoque y el enfoque basado en el mapeo explícito de las observaciones a partir de una función $\Phi : \mathcal{X} \to \mathcal{H}$ son equivalentes. Es decir, que

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{H}} \tag{8}$$

en donde la correspondencia entre la función Kernel K y el espacio $\mathcal{H}$ es única, en virtud del Teorema de Moore-Aronszajn (1950), y la existencia de una función Φ para cierta función Kernel K está garantizada en virtud del Teorema de Mercer (1909).

2.3.1. Funciones Kernels

Más concretamente, una función Kernel K es una función $K : \mathcal{X} \times \mathcal{X} \to R$ que es continua, simétrica y positiva definida. La noción de función Kernel generaliza la noción de matriz simétrica positiva definida; en particular para una realización finita de los datos (i.e. una muestra) $\mathcal{S}$, se tiene que la matriz Kernel $K_{\mathcal{S}} = [k_{i,j}]$ ha de ser simétrica y positiva definida ya que $k_{i,j} = K(x_i, x_j)$. Para consideraciones prácticas, el hecho de que la matriz Kernel satisfaga esta propiedad implica que el problema de optimización del método SVM es convexo, por lo que el problema tiene siempre tiene solución.

2.3.2. Ejemplos de Kernels

De acuerdo a lo discutido con anterioridad, basta con proponer un Kernel (i.e. una función que satisfaga la condición de continuidad, simetría y positividad) de modo que se sabe que la ecuación 8 se satisface para alguna función Φ, para resolver el problema de clasificación.

Algunos de los Kernels más utilizados en aplicaciones son los siguientes:

- Polinomial homogéneo: $k(x_i, x_j) = (x_i \cdot x_j)^d$ para $d \in N$.
- Polinomial no homogéneo: $k(x_i, x_j) = (x_i \cdot x_j + c)^d$, para $c \in R$ y $d \in N$.
- Tangente hiperbólica: $k(x_i, x_j) = \tanh(\kappa x_i \cdot x_j + c)$, para $\kappa > 0$ y $c < 0$.
- Gaussiano de base radial: $k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, para $\gamma > 0$.

Este último Kernel forma parte de una amplia familia de funciones llamadas *funciones de base radial*, cuyos valores dependen de una distancia entre dos puntos, i.e. $\phi(x, c) = \phi(\|x - c\|)$, y es uno de los más utilizados en aplicaciones.

Además, tiene la ventaja de tener una interpretación inmediata: si dos observaciones x_i y x_j son muy parecidas, entonces $\|x_i - x_j\| \to 0$ y por lo tanto $\exp(-\gamma\|x_i - x_j\|^2) = k(x_i, x_j) \to 1$; mientras que si son muy diferentes, entonces $\|x_i - x_j\| \to \infty$ y por lo tanto $\exp(-\gamma\|x_i - x_j\|^2) = k(x_i, x_j) \to 0$. Es decir, valores del Kernel cercanos a 1 corresponden a observaciones muy similares, mientras que valores cercanos a cero corresponden a observaciones muy diferentes.

2.4. Formulación del método

Históricamente, los primeros algoritmos para reconocimiento de patrones fueron clasificadores lineales. Dadas dos clases $y \in \{-1, 1\}$, una nueva observación x puede ser clasificada de acuerdo a una función lineal $f(x) = w \cdot x + b$ de modo que se tome el signo de esta función para realizar la clasificación. Además, se puede realizar una transformación Φ a las nuevas observaciones de modo que la función discriminante sea $f(x) = w \cdot \Phi(x) + b$, y en donde la transformación Φ puede ser realizada de forma explícita, tal como se menciona en la sección 2.2. El hiperplano obtenido mediante $f(x) = 0$ define una frontera de decisión en el espacio transformado y los parámetros w y b son determinados a partir de la muestra disponible de los datos.

La formulación actual del método SVM se basa en estas ideas, junto con otros tres ingredientes escenciales:

- Hiperplanos óptimos.
- Margen suave.
- El truco del Kernel.

2.4.1. Hiperplanos óptimos

El conjunto de datos es *linealmente separable* si existe una función lineal cuyo signo coincide con la clase a la que pertenecen los datos. Si el conjunto de datos es linealmente separable, entonces existe una infinidad de (hiper)planos separadores.

Para elegir alguno de estos posibles hiperplanos, se puede utilizar la noción de *margen*. Intuitivamente, el margen representa la distancia del hiperplano hacia el punto más cercano de cada una de las dos clases. Por lo tanto, una elección razonable del hiperplano es aquel cuyo margen es máximo. Esta idea es ilustrada en la gráfica 2. El problema de encontrar la función $f(x) = w \cdot \Phi(x) + b$ que maximice el margen puede ser formulado de la siguiente forma.

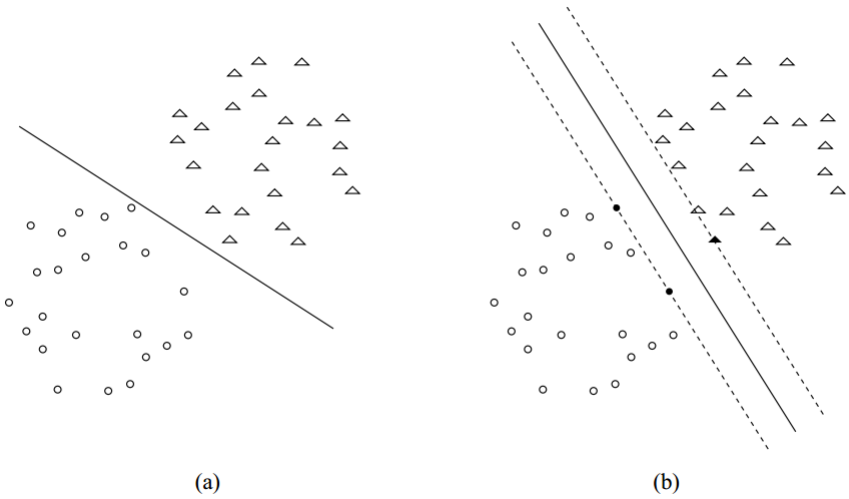

Figura 2: Un conjunto de datos con dos planos de diferente margen. En b) el plano es el de margen máximo y los puntos sombreados son aquellos que definen la función de separación.

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{sujeto a:} \quad & y_i\left(w \cdot \Phi(x_i) + b\right) \geq 1, \quad \forall i \end{aligned} \tag{9}$$

Esta formulación es conocida como la **formulación primal** del problema. Por otra parte, utilizando Teoría de Dualidad, es posible establecer el siguiente problema equivalente, el cual es conocido como la **formulación dual** del problema.

$$\begin{aligned} \max_{\alpha} \quad & \tilde{L}(\alpha) = \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) \\ \text{sujeto a:} \quad & \alpha_i \geq 0 \quad \forall i \\ & \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \forall i \end{aligned} \tag{10}$$

Los coeficientes α_i representan los multiplicadores de Lagrange del problema primal 9. Esta formulación es más sencilla de resolver computacionalmente, ya que las restricciones son más simples. Resolviendo el problema para el vector α y recuperando la solución para el vector w, puede probarse que la solución es de la forma

$$w = \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i) \tag{11}$$

Más aún, al resolver el problema de optimización se obtiene que algunos de los coeficientes α_i son nulos, por lo que la solución sólo depende de un conjunto de los datos. Esta idea también es ilustrada en la gráfica 2; el subconjunto de datos para los cuales $\alpha_i \neq 0$ son aquellos que definen el vector w, y por lo tanto, el hiperplano separador de máximo margen. Los datos que pertenecen a este subconjunto son llamados **vectores de soporte**. Si denotamos a este subconjunto como $SV = \{x_i | \alpha_i \neq 0\}$ entonces la solución puede escribirse como

$$w = \sum_{i \in SV} \alpha_i y_i \Phi(x_i) \tag{12}$$

de modo que para una observación x que es mapeada mediante la transformación Φ, la función de decisión está dada por

$$f(x) = w \cdot \Phi(x) + b = \sum_{i \in SV} \alpha_i y_i \Phi(x_i) \cdot \Phi(x) + b \tag{13}$$

2.4.2. Margen suave

Cuando el conjunto de datos no es linealmente separable, no se puede hablar de la noción de hiperplano óptimo. Sin embargo, este problema puede ser resuelto al permitir que algunos de los datos violen las restricciones del margen que aparecen en la formulación 9. Estas violaciones a las restricciones pueden ser representadas mediante variables de holgura $\xi_i, \forall i = 1, \ldots, n$, y adicionalmente un parámetro C controla el *tradeoff* que existe entre el tamaño del margen y el número de violaciones a las restricciones del margen. Esta idea es ilustrada en la gráfica 3. El problema análogo a 9 que relaja las restricciones al margen puede ser formulado de la siguiente forma.

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C \textstyle\sum_{i=1}^{n} \xi_i \\ \text{sujeto a:} \quad & y_i \left(w \cdot \Phi(x_i) + b\right) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned} \tag{14}$$

De forma similar, la formulación dual del problema análoga a 10, está dada por

$$\begin{aligned} \max_{\alpha} \quad & \tilde{L}(\alpha) = \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) \\ & \text{sujeto a:} \quad 0 \leq \alpha_i \leq C \quad \forall i \\ & \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \forall i \end{aligned} \tag{15}$$

en donde ahora aparece el parámetro C como una cota superior para los parámetros α_i. Este problema de optimización tiene como función objetivo una función cuadrática con restricciones lineales, de modo que representa un problema de programación cuadrática.

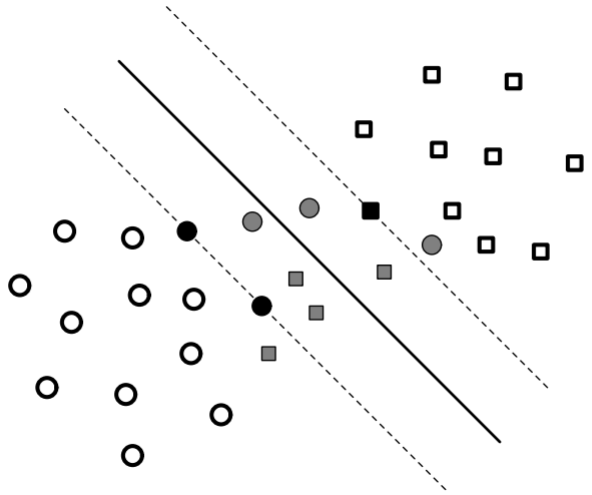

Figura 3: Un conjunto de datos que no es linealmente separable. El problema puede ser resuelto al permitir que algunos de los datos violen las restricciones del margen.

2.4.3. El truco del Kernel

El **truco del Kernel** simplifica el problema de optimización del método SVM. El *truco* consiste en reemplazar el producto interno de los vectores de observaciones en el espacio transformado por evaluaciones mediante un Kernel de los vectores de observaciones en el espacio original. La función objetivo de la formulación dual del problema de optimización presentado anteriormente está dado, para datos sin transformar, por

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{16}$$

de modo que se puede sustituir el producto entre vectores $x_i \cdot x_j = \langle x_i, x_j \rangle_{\mathcal{X}}$ por el producto interno $\langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{H}}$ para cierta función $\Phi : \mathcal{X} \to \mathcal{H}$, es decir

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{H}} \tag{17}$$

de forma que el problema sea resuelto en el espacio transformado; es decir, la separación lineal es llevada a cabo a través de las variables transformadas mediante la función Φ. Sin embargo, de acuerdo a lo discutido sobre el mapeo implícito a través de Kernels, se puede utilizar una función Kernel K que satisface, como ya se ha mencionado, $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{H}}$ para cierta

función Φ, por lo que la función objetivo *kernelizada* es

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{18}$$

Lo anterior implica que las evaluaciones $\Phi(x_i)$ no necesitan realizarse, ni tampoco el producto interno entre las evaluaciones. Es decir, a través de las evaluaciones $K(x_i, x_j)$ mediante el Kernel, tal producto interno es calculado implícitamente y computacionalmente la resolución del problema de optimización no requiere mayores cálculos respecto de la formulación sin kernelizar 16.

Por otra parte, se ha visto que la solución del problema está dada por

$$f(x) = w \cdot x + b = \sum_{i \in SV} \alpha_i y_i x_i \cdot x + b \tag{19}$$

de modo que con los datos transformados, la solución está dada por

$$f(x) = w \cdot \Phi(x) + b = \sum_{i \in SV} \alpha_i y_i \langle \Phi(x_i), \Phi(x) \rangle_{\mathcal{H}} + b \tag{20}$$

y debido a que $\langle \Phi(x_i), \Phi(x) \rangle_{\mathcal{H}} = K(x_i, x)$, la solución *kernelizada* es

$$f(x) = \sum_{i \in SV} \alpha_i y_i K(x_i, x) + b \tag{21}$$

por lo que la solución sólo depende de las evaluaciones a través del Kernel entre la nueva observación x y un subconjunto de los datos (los vectores de soporte).

2.5. Software

Algunas de las implementaciones del método más utilizadas son las siguientes.

- **libSVM** (C++).
- **SVMlight** (C++).
- **kernlab** (R).
- **e1074** (R).
- **sklearn** (Python).

Por otra parte, dado que el problema de optimización asociado al método es uno de programación cuadrática, es posible utilizar cualquier *solver* para este tipo de problemas para obtener una solución. Por ejemplo, en **SAS IML** se puede utilizar el módulo **lcp**. El *software* utilizado para el proceso de modelización en este desarrollo fue **R** utilizando principalmente el paquete **kernlab**, mientras que la implementación del producto final se realizó a través de **SAS**.

3. Regresión Logística

Para estimar la probabilidad de incumplimiento se define el indicador de incumplimiento como variable dependiente y como variables independientes, variables pertenecientes al módulo informacional. Como la variable dependiente es binaria, se procede a realizar una estimación utilizando un modelo de regresión logística.

La especificación del modelo está dada por:

$$P\left(y_i = 1|x_i, \beta\right) = \frac{exp(x_i'\beta)}{1 + exp(x_i'\beta)} = \frac{1}{1 + exp(-x_i'\beta)}$$

en donde y_i refiere al indicador de incumplimiento, x_i al conjunto de variables independientes y β a los parámetros que van a ser estimados por máxima verosimilitud.

El valor esperado del indicador de incumplimiento es simplemente la probabilidad que $y_i = 1$:

$$E(\, y_i \mid x_i, \beta) = P\left(y_i|x_i, \beta\right) \tag{22}$$

Esta especificación de la media condicionada implica que el modelo de regresión puede ser reescrito como:

$$y_i = 1 - \frac{exp(-x_i'\beta)}{1 + exp(-x_i'\beta)} + \epsilon_i \tag{23}$$

Donde ϵ_i es un residuo que representa el desvío de y_i respecto a la media condicionada, y la media y la varianza condicionada están dadas por:

$$E(\, \epsilon_i \mid x_i, \beta) = 0 \tag{24}$$

$$var(\, \epsilon_i \mid x_i, \beta) = \left(\frac{exp(-x_i'\beta)}{1 + exp(-x_i'\beta)}\right)\left(1 - \frac{exp(-x_i'\beta)}{1 + exp(-x_i'\beta)}\right) \tag{25}$$

La interpretación de los valores de los coeficientes es complicada por el hecho de que los coeficientes estimados en un modelo binario no pueden ser interpretados como el efecto marginal sobre la variable dependiente. El efecto marginal de x_j sobre la probabilidad condicional viene dada por:

$$\frac{\partial E(\, y_i \mid x_i, \beta)}{\partial x_{ij}} = \left(\frac{-exp(-x_i'\beta)(1 + 2exp(-x_i'\beta))}{(1 + exp(-x_i'\beta))^2}\right)\beta_j \tag{26}$$

Vale recalcar que β_j está ponderado por el factor que se menciona arriba, que depende de los valores de todos los regresores en x. La dirección del efecto de un cambio en x_j depende solamente del signo del coeficiente β_j. Los valores positivos de β_j implican que incrementar x_j aumentará la probabilidad de incumplimiento, mientras que los valores negativos lo contrario.

Por otra parte, se logra una interpretación alternativa de los coeficientes observando que los ratios de los coeficientes dan una medida de los cambios relativos en las probabilidades:

$$\frac{\frac{\partial E(y_i|x_i,\beta)}{\partial x_{ij}}}{\frac{\partial E(y_i|x_i,\beta)}{\partial x_{ik}}} = \frac{\left(\frac{-exp(-x_i'\beta)(1+2exp(-x_i'\beta))}{(1+exp(-x_i'\beta))^2}\right)\beta_j}{\left(\frac{-exp(-x_i'\beta)(1+2exp(-x_i'\beta))}{(1+exp(-x_i'\beta))^2}\right)\beta_k} \tag{27}$$

4. Descripción de procesos

4.1. Aprovisionamiento de información

El aprovisionamiento, el tratamiento inicial de los datos, la marca de incumplimiento y la segmentación fueron proporcionadas por el área de ASRM, al igual que en el desarrollo Gravity.

4.2. Segmentación

La siguiente figura ilustra la segmentación utilizada en el desarrollo, la cual es la misma que en el desarrollo Gravity.

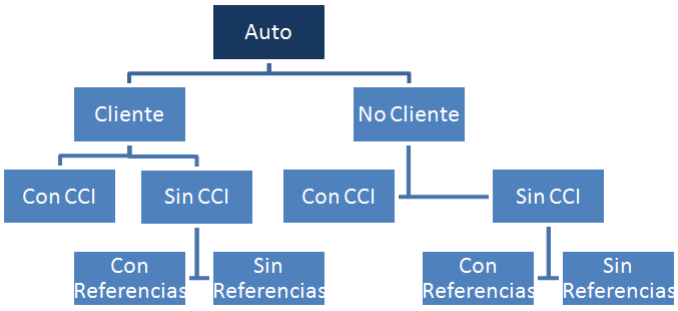

Figura 4: Ilustración de los segmentos utilizados en el desarrollo de los modelos

Los segmentos se definen como se describe a continuación.

1. Clientes Bancomer con CCI
2. Clientes Bancomer sin CCI, con Referencias en Buró
3. Clientes Bancomer sin CCI, sin Referencias en Buró
4. No Clientes con CCI
5. No Clientes sin CCI, con referencias
6. No Clientes sin CCI, sin referencias

Por otra parte, se define a un cliente como aquel cuyo ingreso MILA (YKCP) reportado sea superior a los $6000,00 a la fecha de solicitud.

4.3. Muestras de desarrollo y validación

El conjunto de datos se divide en una muestra de entrenamiento y una muestra de prueba. Para esto, se realizó muestreo estratificado para conservar las tasas de incumplimiento muy similares entre las muestras de entrenamiento y prueba para cada segmento. Se efectuó una partición en donde 80 % de las observaciones forman parte de la muestra de entrenamiento (o de desarrollo) y el restante 20 % de la muestra de prueba (o de validación).

4.4. Variables

El conjunto de variables susceptibles de ser incluidas en los modelos se enlistan a continuación.

1. BC_SCORE
2. CCI
3. Cuentas_Buro
4. Cuentas_Cierre
5. Edad_PF
6. LTV
7. MaxCredMax
8. MaxLimCred
9. MaxSdoActual
10. MaxSdoVencido
11. MontoSolicitado_PF

12. Re_Antig_Cliente
13. Re_Antig_Empleo
14. Re_Antig_Hogar
15. Re_Edad
16. Re_Enganche
17. Re_Max_Antig_Cuentas_Ab
18. Re_Max_Antig_Cuentas_NoRev
19. Re_Max_Mop_24
20. Re_Max_Mop_Actual
21. Re_Nivel_Estudios
22. Re_Num_Consultas_12meses
23. Re_Num_Cuentas_Ab
24. Re_Plazo
25. Re_Ratio_Endeudamiento
26. Re_Tasa_Incidencias_12
27. Re_Tasa_Uso_Cuentas_Rev
28. Re_Tipo_Contrato_Lab
29. SumCredMax
30. SumLimCred
31. SumSdoActual
32. SumSdoVencido
33. mensualidad

El prefijo **Re**_ hace referencia a variables que originalmente forman parte del *Score Reactivo*. En los siguientes cuadros se muestran las variables utilizadas en los modelos para cada segmento.

SEGMENTO 1
BC_SCORE
CCI
Cuentas_Buro
LTV
MaxSdoActual
Re_Antig_Cliente
Re_Antig_Hogar
Re_Enganche
Re_Max_Antig_Cuentas_Ab
Re_Max_Antig_Cuentas_NoRev
Re_Nivel_Estudios
Re_Num_Consultas_12meses
Re_Plazo
Re_Ratio_Endeudamiento
mensualidad

SEGMENTO 4
BC_SCORE
CCI
Cuentas_Buro
LTV
MaxSdoActual
Re_Antig_Cliente
Re_Antig_Empleo
Re_Antig_Hogar
Re_Enganche
Re_Max_Antig_Cuentas_Ab
Re_Max_Antig_Cuentas_NoRev
Re_Nivel_Estudios
Re_Num_Consultas_12meses
Re_Num_Cuentas_Ab
Re_Plazo
Re_Ratio_Endeudamiento
Re_Tasa_Incidencias_12
Re_Tipo_Contrato_Lab
mensualidad

SEGMENTO 5
LTV
Re_Antig_Cliente
Re_Antig_Hogar
Re_Max_Antig_Cuentas_Ab
Re_Nivel_Estudios
Re_Num_Consultas_12meses
Re_Plazo
mensualidad

SEGMENTO 6
LTV
Re_Antig_Cliente
Re_Antig_Empleo
Re_Antig_Hogar
Re_Nivel_Estudios
Re_Plazo
Re_Tipo_Contrato_Lab
mensualidad

SEGMENTO 2
LTV
Re_Antig_Cliente
Re_Max_Antig_Cuentas_Ab
Re_Num_Consultas_12meses
Re_Plazo
Re_Tasa_Incidencias_12
Re_Tipo_Contrato_Lab
mensualidad

Figura 5: Lista de las variables en los modelos por segmento

4.5. Selección de modelos y validación

4.5.1. Balanceo de clases

La proporción real en la muestra total de los casos etiquetados como malos es baja en cada segmento. Dado que se desea que el modelo sea capaz de discriminar especialmente a los casos malos, se ha decidido realizar un balanceo de las clases, de modo que la proporción sea de 1:1. Sin embargo, se desea que la Probabilidad de Default refleje la proporción real de los casos malos y los casos buenos del universo de clientes para los que se realizó la modelización, por lo que se ha seguido el siguiente enfoque.

- Obtener un modelo de SVM a partir de una muestra balanceada de los datos, de modo que se obtenga un *score* que proporcione un ordenamiento a las observaciones.
- Ajustar una curva de PD paramétrica como función del *score* obtenido, basada en la proporción empírica de casos malos y buenos.

Definimos los siguientes conjuntos. Los datos de entrenamiento están dados por el conjunto

$$\mathcal{S} = \{x_i, y_i\}_{i=1}^{n}, \quad x_i \in \mathcal{X}, y_i \in \mathcal{Y} \tag{28}$$

en donde $\mathcal{X}$ es el espacio de las características de los clientes y $\mathcal{Y} = \{-1, 1\}$ representa la etiqueta de clase de los clientes, mientras que los conjuntos de clientes *buenos* y *malos* se definen como

$$\text{buenos} = \{(x_i, y_i) \in \mathcal{S} | y_i = -1\}, \quad \text{malos} = \{(x_i, y_i) \in \mathcal{S} | y_i = 1\} \tag{29}$$

4.5.2. Selección de Kernel y parámetros

En la formulación del método SVM, tal como se ha discutido en la sección 2.4.2, se tiene un parámetro C cuyo objetivo es controlar el *tradeoff* que existe entre el tamaño del margen y el número de violaciones a las restricciones del margen. Por otra parte, dependiendo de la función Kernel utilizada, se tiene otro conjunto de parámetros que han de seleccionarse. En esta aplicación, se ha decidido utilizar el Kernel Radial debido a que es fácilmente intrpretable en términos de similitud y a que ha mostrado ser de los más eficientes en términos de predicción en numerosas aplicaciones en otros campos. Este Kernel contiene sólo un parámetro (el parámetro σ) por seleccionar. De este modo, la selección (o *calibración*) de los parámetros se restringe a una búsqueda en dos dimensiones para la pareja de parámetros (C, σ). Esta búsqueda se ha realizado de una forma tradicional, combinando las estretegias de **search grid** y **validación cruzada** de modo que se elige la pareja de parámetros en donde el AUC es máximo.

La búsqueda de los parámetros se realizó sobre el siguiente conjunto, el cual fue determinado después de realizar pruebas sobre diversos conjuntos y refinamientos del *grid*:

$$(C, \sigma) \in \{1, 10, 100\} \times \{.1, .3, ,5, 1, 10, 100\} = \mathcal{P} \tag{30}$$

de forma que $|\mathcal{P}| = 18$.

4.5.3. Selección del modelo

Dado el bajo número de casos etiquetados como malos y al balanceo de la proporción de las clases que se desea realizar, es necesario seguir una estrategia de *remuestreo* de forma repetitiva un número suficientemente grande de veces. Esto es porque con solo una muestra de los datos disponibles se obtendría un modelo sobreajustado. Debe notarse que, a diferencia de otros métodos clásicos como la regresión logística, el método SVM es un método *basado en casos (instance-based learning)*. En el método SVM, el conjunto de parámetros que se obtiene corresponde a las observaciones y no a las variables con las que se cuenta; por lo tanto no se puede hablar de encontrar un *promedio* de los parámetros del modelo a través de remuestreo de forma repetitiva. De este modo, al seguir el enfoque de remuestreo con clases balanceadas, el problema se traduce en encontrar el mejor conjunto de parámetros (i.e. el mejor conjunto de *vectores de soporte* y sus coeficientes), en términos de poder de predicción.

La idea de la estrategia es la siguiente. En cada iteración, seleccionar una muestra reducida y balanceada de casos buenos y malos. Para esta muestra, obtener un modelo de SVM, habiendo elegido los parámetros mediante *search grid* y *validación cruzada*. Para este modelo, obtener una medida de desempeño honesta (el AUC o el Gini) sobre una muestra independiente cuya proporción de casos buenos y malos sea muy similar a la real (la muestra original exceptuando los casos que se utilizaron para obtener el modelo). De esta forma, repetir este procedimiento un numero suficiente de veces de modo que se pueda elegir un modelo cuyo desempeño sea suficientemente bueno (relativo al promedio

del desempeño sobre todas las iteraciones) en muestras de entrenamiento y de validación.

Más concretamente, el procedimiento realizado se ilustra en el siguiente seudocódigo.

Data: Muestra de los datos: $\mathcal{S}_0 = \{x_i, y_i\}_{i=1}^n$.
input : Número máximo de iteraciones T, tamaño de las submuestras m, número de capas de validación cruzada K.
output: Conjuntos de parámetros $\{(\alpha_i)_1^n\}_{t=1}^T$, vectores de medidas de desempeño, intervalos de confianza y conjunto de parejas de parámetros óptimos:

$$(AUC^{VC})_{t=1}^T, \quad (AUC^{Test})_{t=1}^T, \quad (C^*, \sigma^*)_{t=1}^T$$

Inicializar $t \leftarrow 1$;
while $t \leq T$ **do**
  Imponer una semilla, $seed_t = t$;
  Seleccionar de forma seudo-aleatoria una muestra balanceada $\mathcal{S}_t = \{x_j, y_j\}_{j=1}^m$ de tamaño $m < n$;
  Efectuar *search grid* con *validación cruzada* de K capas para obtener la pareja de parámetros $(C^*, \sigma^*)_t$;
  Construir un modelo de SVM con los parámetros $(C^*, \sigma^*)_t$ sobre la muestra S_t ;
  Evaluar el desempeño del modelo sobre el conjunto de datos $\mathcal{S}_0 \setminus \mathcal{S}_t$;
  $t \leftarrow t + 1$;
end

Los valores utilizados en el desarrollo del modelo fueron $T = 125, m = 2\lfloor \frac{|malos|}{4} \rfloor$ (i.e. hay $\frac{m}{4}$ casos buenos y $\frac{m}{4}$ casos malos en cada submuestra), $K = 3$. La semilla en cada iteración es impuesta para la replicación de los modelos seleccionados como candidatos.

4.5.4. Interpretación del *score*

La función de predicción (el *score*) del modelo con el Kernel radial utilizado es

$$f(x) = \sum_{i \in SV} \alpha_i y_i \exp^{-\gamma \|x - x_i\|^2} + b \tag{31}$$

En el contexto de esta aplicación, las observaciones x_i representan clientes cuyas variables provienen de información interna como del Buró de crédito.

Para obtener el *score* de un nuevo cliente, los valores de sus variables son comparados con los de otros clientes (los que corresponden a los vectores de soporte), clasificados previamente como solventes o insolventes. De acuerdo a 31, se puede decir que el *score* es obtenido como una suma ponderada de medidas de similitud entre un nuevo cliente y un subconjunto de los clientes utilizados en el desarrollo, los cuales se sabe previamente son solventes o insolventes. Por lo tanto, el *score* de un cliente depende de con qué grupo tiene mayor similitud, y a qué grado.

Para ilustrar esta idea, considérese el siguiente ejemplo y la gráfica 6. Supóngase que se tienen un conjunto de clientes buenos y malos (en color azul y rojo, respectivamente) y que el modelo sólo depende de dos variables (el ratio financiero 1 y el ratio financiero 2). Supóngase también que el problema es linealmente separable en estas dos variables. En la gráfica se ilustra la función de clasificación (el *score*) y los vectores de soporte, las cuales corresponden a las observaciones de las cuales depende el score. Entonces de acuerdo a 31 y a lo ilustrado en la gráfica, el cálculo del *score* para un nuevo cliente x puede ser explicado como

$$\textbf{score}(x) = \sum_{i=1}^{3} \alpha_i \textbf{similitud}(malo_i, x) + \sum_{j=1}^{2} \beta_j \textbf{similitud}(bueno_j, x) \tag{32}$$

en donde los parámetros α_i y β_j son las ponderaciones correspondientes a los clientes malos y buenos, respectivamente, y los cuales son encontrados mediante el método. Este mismo razonamiento puede ser extrapolado al caso en donde se realiza el mapeo implícito de las observaciones mediante un Kernel, de modo que la separación lineal tome lugar en un espacio de mayor dimensión.

4.6. Mapeo a probabilidades

La función de predicción obtenida mediante el método

$$f(x) = \sum_{i \in SV} \alpha_i y_i K(x_i, x) + b \tag{33}$$

representa por si misma una función que da un ordenamiento a nuevas observaciones. Es decir, representa un *score*. La función de predicción de clase está dada por $g(x) = sgn(f(x))$; sin embargo, para el propósito de la aplicación

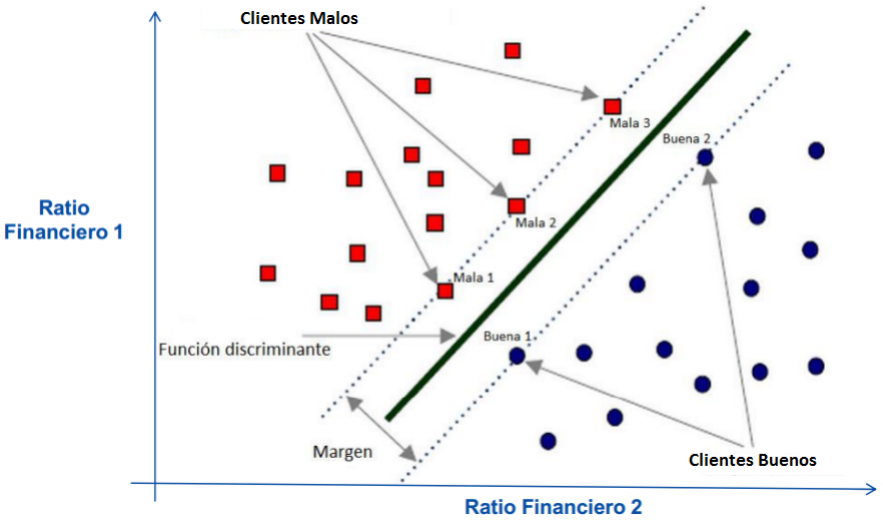

Figura 6: Interpretación del *score* generado por el método SVM. El *score* de un cliente depende de con qué grupo tiene mayor similitud.

deseada, no se requiere de la predicción de clase sino precisamente de un ordenamiento. Con este *score*, es posible tener medidas de desempeño como el Gini y el área bajo la curva ROC (AUC) para evaluar la capacidad de discriminación del modelo.

Por otra parte, al contar con este *score*, se puede ajustar una curva paramétrica de Probabilidad de Default de la forma tradicional a través de las tasas de mora empíricas. De esta forma se impone la forma funcional para la curva de PD:

$$PD(f) = \frac{1}{1 + \exp(\beta f + \beta_0)} \tag{34}$$

la cual se ajusta de acuerdo a los datos observados para el *score* f.

4.7. Resumen de los modelos SVM

En este apartado se presenta un resumen de los modelos obtenidos para los segmentos 1 y 4. Se presentan los parámetros de la función de decisión, así como los parámetros de la curva de PD y las estimaciones del desempeño de los modelos.

Las secciones 7.3 y 7.4 contienen el código utilizado para calificar nuevas observaciones para cada segmento, respectivamente. Los archivos que contienen los vectores de soporte, así como las bases de entrenamiento y de prueba para poder replicar el código se encuentran en la carpeta **Modelos SVM**.

4.7.1. Segmento 1

Parámetros del modelo SVM:

$$C = 1 \quad \sigma = 0{,}1 \quad b = 0{,}09309717 \quad |SV| = 2635$$

Parámetros de la curva de PD:

$$\beta = -1{,}744097 \quad \beta_0 = 0{,}02370129$$

Estimaciones del desempeño por validación cruzada y en muestra de prueba:

$$\text{AUC}^{VC} = ,8778 \quad \text{AUC}^{Test} = ,8590$$

4.7.2. Segmento 4

Parámetros del modelo SVM:

$$C = 1 \quad \sigma = 0{,}1 \quad b = 0{,}1106492 \quad |SV| = 490$$

Parámetros de la curva de PD:

$$\beta = -1{,}518724 \quad \beta_0 = -0{,}09612522$$

Estimaciones del desempeño por validación cruzada y en muestra de prueba:

$$\text{AUC}^{VC} = ,8377 \quad \text{AUC}^{Test} = ,8283$$

4.7.3. Comparación con modelos logísticos

Adicionalmente se realizó el ajuste de modelos logísticos para ambos segmentos con la finalidad de comparar su desempeño con el obtenido mediante los modelos SVM. Para esto, se utilizaron las mismas variables que contienen los modelos SVM para cada segmento, respectivamente. Por otra parte, se siguió una estrategia estándar: construir el modelo a partir de labase de entrenamiento y probar su desempeño en la base de prueba. El código utilizado se puede consultar en la sección 7.5 y los archivos necesarios para replicar el código se encuentran en la carpeta **Regresiones**.

Segmento 1

Estimaciones del desempeño por validación cruzada y en muestra de prueba:

$$\text{AUC}^{Train} = ,8405 \quad \text{AUC}^{Test} = ,8089$$

Segmento 4

Estimaciones del desempeño por validación cruzada y en muestra de prueba:

$$\text{AUC}^{Train} = ,7793 \quad \text{AUC}^{Test} = ,7861$$

```
Call:
glm(formula = Y2 ~ ., family = binomial, data = seg1_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5456  -0.2626  -0.1653  -0.1074   3.7062

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                -6.824e+02  1.023e+02  -6.667 2.61e-11 ***
BC_SCORE                   -1.381e-02  4.171e-04 -33.110  < 2e-16 ***
CCI                        -2.156e-01  1.446e-02 -14.913  < 2e-16 ***
Cuentas_Buro               -9.195e-03  3.714e-03  -2.476 0.013303 *
MaxSdoActual               -3.227e-07  9.454e-08  -3.414 0.000640 ***
LTV                        -5.380e-03  1.121e-02  -0.480 0.631247
mensualidad                 1.823e-05  9.003e-06   2.024 0.042927 *
Re_Antig_Cliente           -1.855e-02  4.087e-03  -4.538 5.68e-06 ***
Re_Antig_Hogar             -7.783e-03  2.318e-03  -3.358 0.000786 ***
Re_Enganche                 6.882e+00  1.024e+00   6.718 1.84e-11 ***
Re_Max_Antig_Cuentas_Ab    -2.545e-03  5.964e-04  -4.268 1.97e-05 ***
Re_Max_Antig_Cuentas_NoRev  2.458e-03  9.635e-04   2.551 0.010748 *
Re_Nivel_Estudios          -1.846e-02  1.950e-02  -0.947 0.343769
Re_Plazo                    4.249e-03  2.431e-03   1.748 0.080537 .
Re_Ratio_Endeudamiento      6.909e+00  1.024e+00   6.749 1.49e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figura 7: Resumen modelo logístico Segmento 1

```
Call:
glm(formula = Y2 ~ ., family = binomial, data = seg4_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9857  -0.4546  -0.3281  -0.2314   3.2349

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                -3.537e+02  4.645e+01  -7.614 2.66e-14 ***
BC_SCORE                   -7.781e-03  2.196e-04 -35.427  < 2e-16 ***
CCI                        -1.367e-01  7.675e-03 -17.807  < 2e-16 ***
Cuentas_Buro                5.086e-04  2.072e-03   0.245  0.80612
MaxSdoActual                5.784e-08  2.775e-08   2.084  0.03714 *
LTV                         3.231e-03  5.330e-03   0.606  0.54434
mensualidad                 4.179e-05  4.345e-06   9.618  < 2e-16 ***
Re_Antig_Cliente           -7.363e-03  2.278e-03  -3.232  0.00123 **
Re_Antig_Empleo            -3.963e-03  1.523e-03  -2.603  0.00924 **
Re_Antig_Hogar             -4.665e-03  1.141e-03  -4.088 4.36e-05 ***
Re_Enganche                 3.556e+00  4.649e-01   7.647 2.06e-14 ***
Re_Max_Antig_Cuentas_Ab    -7.999e-04  3.018e-04  -2.650  0.00804 **
Re_Max_Antig_Cuentas_NoRev  4.256e-04  5.062e-04   0.841  0.40041
Re_Nivel_Estudios          -5.132e-02  8.400e-03  -6.109 1.00e-09 ***
Re_Num_Cuentas_Ab          -4.175e-02  7.529e-03  -5.546 2.92e-08 ***
Re_Plazo                    9.063e-03  1.192e-03   7.600 2.95e-14 ***
Re_Ratio_Endeudamiento      3.571e+00  4.647e-01   7.685 1.53e-14 ***
Re_Tasa_Incidencias_12      1.425e+00  5.367e-02  26.555  < 2e-16 ***
Re_Tipo_Contrato_Lab       -5.020e-03  3.630e-03  -1.383  0.16673
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figura 8: Resumen modelo logístico Segmento 4

4.8. Resumen de los modelos Regresión Logística

4.8.1. Análisis bivariante y trameados

La información del análisis bivariante para las variables que quedaron incluidas en los modelos de los segmentos 5, 6, 2, así como la información de los tramos y sus respectivas WoE's se encuentran en el archivo **Tablas_bivariados.xlsx**.

4.8.2. Regresiones

Los detalles de las regresiones logísticas con tramos de las variables para los modelos de los segmentos 5, 6, 2, se encuentran en el archivo **Regresiones.xlsx**.

4.8.3. Resultados

Los siguientes cuadros presentan un resumen del desempeño de los modelos por segmento, en términos de AUC y coeficiente de Gini. Para cada caso se muestra el desempeño en la base de entrenamiento, en la base de prueba, y en la base completa, así como información sobre el volumen de las muestras y la proporción de casos buenos y malos.

SEGMENTO 5: ENTRENAMIENTO		
	AUC	GINI
Logistic Score	**0.84**	**0.68**
Gravity	0.786	0.572

SEGMENTO 5: PRUEBA		
	AUC	GINI
Logistic Score	**0.838**	**0.676**
Gravity	0.798	0.596

SEGMENTO 5: FULL		
	AUC	GINI
Logistic Scor	**0.837**	**0.674**
Gravity	0.79	0.58

Y2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	5710	97.31	5710	97.31
1	158	2.69	5868	100

Y2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1427	97.34	1427	97.34
1	39	2.66	1466	100

Figura 9: Resultados Segmento 5

SEGMENTO 6: ENTRENAMIENTO		
	AUC	GINI
Logistic Score	**0.81**	**0.62**
Gravity	0.731	0.462

SEGMENTO 6: PRUEBA		
	AUC	GINI
Logistic Score	**0.744**	**0.488**
Gravity	0.737	0.474

SEGMENTO 6: FULL		
	AUC	GINI
Logistic Score	**0.796**	**0.592**
Gravity	0.732	0.464

Y2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	3187	92.38	3187	92.38
1	263	7.62	3450	100

Y2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	796	92.45	796	92.45
1	65	7.55	861	100

Figura 10: Resultados Segmento 6

SEGMENTO 2: ENTRENAMIENTO		
	AUC	GINI
Logistic Score	0.941	0.882
Gravity	0.805	0.61

*En este segmento no se hizo partición Train/Test por el volumen de datos y por el número muy bajo de malos.

Y2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	3030	98.83	3030	98.83
1	36	1.17	3066	100

Figura 11: Resultados Segmento 2

5. Implementación en SAS

En los siguientes apartados se describe a detalle la **macro** de SAS programada para realizar la implementación del método SVM. El código puede ser consultado en la sección 7.1.

5.1. Macro PROC_SVM

- **input**: *(data, datatest, labels, cost, sigma)*

  1. *data*: Es la base de entrenamiento o desarrollo. Debe contener las variables a utilizar y la variable binaria a predecir. Las variables explicativas deben encontrarse estandarizadas.
  2. *datatest*: Es la base de prueba o validación. Debe contener las variables a utilizar y la variable binaria a predecir. Las variables explicativas deben estar estandarizadas de acuerdo a las medias y desviaciones de la base de entrenamiento. Nótese que esta base puede ser la misma de entrenamiento, si es que se desea evaluar el desempeño sobre la misma o si es que no se cuenta con una base de prueba.
  3. *labels*: Es el nombre de la columna que contiene la variable binaria a predecir. Debe encontrarse en formato numérico, tomando valores de -1 o 1 para las clases.
  4. *cost*: Es el parámetro de costo de la formulación dual del problema de SVM.
  5. *sigma*: Es el parámetro del Kernel radial.

- Utilizar IML para plantear el problema dual de SVM de forma matricial como uno de programación cuadrática sobre la base de entrenamiento.
- Resolver el problema con el módulo LCP de IML. Recuperar la solución y construir la función de predicción.
- Evaluar el desempeño del modelo sobre la base de prueba.
- **Output**: *(output, outputsvm, outputscore, datatest)*

1. *output*: Tabla que contiene por columnas, para cada observación de la muestra de entrenamiento, los valores: $\alpha_i, y_i, \alpha_i y_i$, $i = 1, \ldots, n$. Nótese que de aquí se obtiene la información de las observaciones que corresponden a los vectores de soporte (aquellas cuya $\alpha_i \neq 0$).
2. *outputsvm*: Renglón que contiene la información: *(auc, numsv, objval)*; es decir, el área bajo la curva ROC calculada sobre la base de prueba, el número de vectores de soporte, y el valor de la función objetivo de la formulación dual del problema de SVM, respectivamente.
3. *outputscore*: Columna que contiene el valor de la función (*score*) $f(x)$ para cada observación de la muestra de prueba, calculada mediante los vectores de soporte resultantes de la muestra de entrenamiento.
4. *datatest*: Es la misma base de prueba recibida como argumento, añadiendo la columna *score_svm* que contiene al vector columna *outputscore*.

6. Conclusiones

El material presentado en este documento ilustra cómo el método SVM se ha utilizado para construir una función *score* para discriminar clientes solventes e insolventes. Se ha discutido cómo el *score* puede ser interpretado en términos de similitud de las características de los clientes evaluados, y cómo puede ser mapeado a una Probabilidad de Default que refleje la tasa de mora del universo modelizado.

7. Códigos

7.1. Macro PROC_SVM

```
/* ********************************************************************************** */
/* ********************************** PROC SVM  ********************************** */
/* ********************************************************************************** */

%macro proc_svm(data, datatest, labels, cost, sigma);

/* ARGUMENTOS: */
/* data: muestra de entrenamiento */
/* datatest: muestra de prueba */
/* labels: columna con las etiquetas de clase (1 o -1) */
/* cost: parametro de costo */
/* sigma: parametro del Kernel RBF */

proc contents data=&data.(drop=&labels.) out=names(keep=name) noprint;
run;

proc sql;
    select name into :independ separated by " "
    from names;
quit;

proc iml;
USE &data; /* OBJETO DATASET A OBJETO MATRIX (MUESTRA DE ENTRENAMIENTO) */
READ ALL VAR{&labels.} INTO y; /* COLUMNA DE LA VARIABLE DEPENDIENTE */
READ ALL VAR{&independ.} INTO x; /* COLUMNAS DE LAS VARIABLES EXPLICATIVAS */
CLOSE &data; /* CERRAR DATASET */

USE &datatest; /* OBJETO DATASET A OBJETO MATRIX (MUESTRA DE PRUEBA) */
READ ALL VAR{&independ.} INTO xtest; /* COLUMNAS DE LAS VARIABLES EXPLICATIVAS */
READ ALL VAR{&labels.} INTO ytest;
CLOSE &datatest; /* CERRAR DATASET */

N = NROW(x); /* NUMERO DE OBSERVACIONES */
H = I(N); /* INICIALIZAR MATRIZ H */

start auc(x, y); /* FUNCION QUE CALCULA EL AUC, DADO UN SCORE Y LAS ETIQUETAS */

   u = unique(x);
   cutpts = (u[1]-1) || u;
    nc = ncol(cutpts);
   TP = j(nc,1,.);   TN = j(nc,1,.); /** allocate **/
```

```
   /** count how many obs greater than and less
      than each cutpoint for Y=0 and Y=1 **/
   x0 = x[loc(y=-1)]; Num0 = nrow(x0); /** num in N **/
   x1 = x[loc(y=1)]; Num1 = nrow(x1); /** num in P **/
   do i = 1 to nc; /** proportions for each cutpoint **/
      TN[i] = sum( x0 <= cutpts[i] ) / Num0;
      TP[i] = sum( x1 > cutpts[i] ) / Num1;
   end;


   /** return false pos (=1-TN) and true pos rates
       (1 - specificity) || sensitivity **/
    speci = (1-TN);
    sensit= TP;


    auc = 0;
    do i=1 to (nrow(speci)-1);
        auc = auc + abs(((speci[(i+1)]-(speci[i]))#(sensit[i]+(sensit[i+1])))/2);
    end;
    /*gin = abs(auc#2-1)*/;
    return (auc);

finish;


start matnorm(x, method); /* FUNCION QUE CALCULA LA NORMA-2 DE UN VECTOR */
   s = substr(upcase(method), 1, 4);
   isValid = (s="FROB" | s="L1" | s="LINF");
   if ^isValid then return(.);

   if (s="FROB") then do;
      return( sqrt(x[##]) );
   end;
   else if (s="L1") then do;
      return( max(x[+,]) );
   end;
   else if (s="LINF") then do;
      return( max(x[,+]) );
   end;
finish;

start keval(arg1, arg2); /* FUNCION QUE CALCULA LA NORMA-2 DE UN VECTOR
DE DIFERENCIA DE OBSERVACIONES */
   dif=arg1-arg2;
```

```
   sqnorm = matnorm(dif, "FROB");
   sqnorm2 = sqnorm*sqnorm;
   return(sqnorm2);
finish;



/* LOOP QUE CREA LA MATRIZ H */
/* H_ij = y_i*y_j*K_ij */
DO i=1 TO N;
    DO j=1 TO i;
        if (i ^= j) then do;
        sqnorm2=keval(x[i,], x[j,]);
        H[i,j] = y[i]*y[j]*exp(-&sigma.*sqnorm2);
        H[j,i] = H[i,j];
        end;
    end;
end;


numvars = ncol(x);
c = repeat (-1,N,1);
yt = y‘;
id = I(N);
G = yt // id;
rel = { ’=’};
desig = repeat(’<=’,N,1);
b = 0;
costv = repeat(&cost.,N,1);
signos = rel//desig;
rightside = b//costv;

/* ************************************************************************** */

/*              Routine to solve quadratic programs            */
      /* names: the names of the decision variables                  */
      /* c:  vector of linear coefficients of the objective function */
      /* H:  matrix of quadratic terms in the objective function     */
      /* G:  matrix of constraint coefficients                       */
      /* signos: character array of values: ’<=’ or ’>=’ or ’=’         */
      /* rightside:   right-hand side of constraints                        */
      /* alpha: returns the optimal value of decision variables   */

start qp( names, c, H, G, rel, b, alpha,objval);

      nr=nrow(G);
```

```
      nc=ncol(G);

         /* Put in canonical form */
      rev=(rel='<=');
      adj=(-1 * rev) + ^rev;
      g=adj# G; b = adj # b;
      eq=( rel = '='  );
      if max(eq)=1 then
      do;
         g=g // -(diag(eq)*G)[loc(eq),];
         b=b // -(diag(eq)*b)[loc(eq)];
      end;
      m=(h || -g`) //(g || j(nrow(g),nrow(g),0));
      q=c // -b;

         /* Solve the problem */
      call lcp(rc,w,z,M,q,1.0E-5);

         /* Report the solution */
      reset noname;
      print ( { '*************Solution is optimal**************',
                '**********No solution possible*****************',
                ' ',
                ' ',
                ' ',
                '***********Solution is numerically unstable*****',
                '************Not enough memory******************',
                '***********Number of iterations exceeded********'}[rc+1]);
      reset name;
      alpha=z[1:nc];
      objval=c`*alpha + alpha`*H*alpha/2;
      print ,'Objective Value ' objval,
             'Decision Variables ' alpha,
             '*********************************************';
  create obj from objval; append from objval;
      finish qp;

   run qp(names,c,H,G,signos,rightside,alpha,objval);

    out = alpha || y || alpha#y;
    create output from out; append from out;

/* ------------------------------------------------------------------------------ */

/* RECUPERANDO LOS VECTORES DE SOPORTE A PARTIR DE LA SOLUCION DEL PROBLEMA DUAL */
```

```
svindex = loc( alpha ^= 0 );
sv = x[svindex,];
svalpha = out[svindex, 3];
svy = out[svindex, 2];

/* INICIALIZAR VECTOR DE SCORES */
L = NROW(xtest);
numsv = NROW(sv);
f = repeat(0,L,1);

/* CONSTRUYENDO LOS SCORES A PARTIR DE LOS VECTORES DE SOPORTE */
DO i=1 TO L;
    DO j=1 TO numsv;
         sqnorm2=keval(xtest[i,], sv[j,]);
        f[i] = f[i] + svalpha[j]*exp(-&sigma.*sqnorm2);
    end;
end;

/* CONSTRUCCION DEL INTERCEPTO b A PARTIR DE LAS C.P.O. DEL PROBLEMA PRIMAL */
b=0;
sumaux=0;

DO i=1 to numsv;
    DO j=1 to numsv;
        sqnorm2= keval(sv[i,], sv[j,]);
        sumaux = sumaux + svalpha[j]*exp(-&sigma.*sqnorm2);
    end;
    b = b -svy[i] + sumaux;
    sumaux=0;
end;

b=b/numsv;

/* SOLUCION FINAL */
f = f-b;

create outputscore from f; append from f;
/* print f;*/
/* CALCULANDO EL AUC PARA LAS OBSERVACIONES EN MUESTRA DE PRUEBA */
auctest = auc(f,ytest);

print objval;
/* RECUPERAMOS Y GUARDAMOS EL AUC, EL NUMERO DE VECTORES DE SOPORTE Y
VALOR DE LA FUNCION OBJETIVO */
outputmatrix = auctest || numsv || objval;
```

```
varNames = {"auc" "numsv" "objval"};

create outputsvm from outputmatrix [colname=varNames]; append from outputmatrix;

data &datatest.;
merge &datatest. outputscore;
rename col1=score_svm;
run;

%mend;
```

7.2. Función *sampling*

```
sampling<-function(base,T){
  library(mlr)
  library(kernlab)
  library(ROCR)
  #library(pROC)
  ################################
  #Argumentos para la funcion MLR
  learner = makeLearner("classif.ksvm",predict.type="prob")
  # SE DEFINE EL CONJUNTO DE VALORES PARA FORMAR EL GRID
  param= makeParamSet(
    makeDiscreteParam("C", values = c(1)),
    makeDiscreteParam("sigma", values = c(.1))
  )
  ctrl = makeTuneControlGrid()
  inner = makeResampleDesc("Holdout")       #Holdout validation
  outer = makeResampleDesc("CV", iters = 3) #K-fold CrossValidation

  ################################
  malos<-subset(base,inc==1)   #Subset con clientes con incumplimiento igual a 1
  num_malos<-nrow(malos)
  buenos<-subset(base,inc==0)  #Subset con clientes con incumplimiento igual a 0

  #En estos vectores vamos a guardar los mejores AUC por VC, los mejores parametros C
  #y sigma en cada iteracion y el AUC con muestra de prueba
  aucs<-rep(0,T)
  Cs<-rep(0,T)
  sigmas<-rep(0,T)
  aucs_train<-rep(0,T)
  ci_lower<-rep(0,T)
  ci_upper<-rep(0,T)
  nsv<-rep(0,T) #Numero de SV
  tprs<-rep(0,T)
  accs<-rep(0,T)

  #Loop:
  # Hacemos T iteraciones. COnservamos los malos
  for (i in 1:T){
    library(mlr)
    set.seed(i)

    ind_samp<-sample(1:nrow(buenos), size=floor(num_malos/4), replace = FALSE, prob = NULL)
    ind_malos<-sample(1:nrow(malos),size=floor(num_malos/4),replace=FALSE,prob=NULL)

    samp<-buenos[ind_samp,]
```

```
    samp_malos<-malos[ind_malos,]
    samp_complemento<-buenos[-ind_samp,]
    samp_malos_complemento<-malos[-ind_malos,]
    base_complemento<-rbind(samp_complemento,samp_malos_complemento)

    basesamp<-rbind(samp,samp_malos)
    sub.base<-basesamp

    sub.base$inc<-as.factor(sub.base$inc)
    task<-makeClassifTask(data=sub.base, target="inc")
    lrnr_task = makeTuneWrapper(learner, resampling = outer, par.set = param,
                                control = ctrl, measure=list(auc,tpr,fpr,acc))
    set.seed(i)
    mod = train(lrnr_task, task)

    aucs[i]<-mod$learner.model$opt.result$y[1]
    tprs[i]<-mod$learner.model$opt.result$y[2]
    accs[i]<-mod$learner.model$opt.result$y[4]
    Cs[i]<-mod$learner.model$opt.result$x[[1]]
    sigmas[i]<-mod$learner.model$opt.result$x[[2]]

    model<-ksvm(inc~., data=sub.base, type = "C-svc", C=mod$learner.model$opt.result$x[[1]],
                kpar=list(sigma = mod$learner.model$opt.result$x[[2]]),prob.model=T)
    nsv[i]<-model@nSV

    set.seed(i)

    preds_train<-predict(model,newdata=base_complemento,type="probabilities")
    preds_train<-preds_train[,2]
    base_complemento$preds<-preds_train

    library(pROC)
    rocobj<-plot.roc(base_complemento$inc, base_complemento$preds,percent=TRUE,
                     ci=TRUE,print.auc=TRUE)
    aucs_train[i]<-rocobj$auc

    detach("package:pROC",unload=T)
    detach("package:mlr",unload=T)

  }
  #Guardamos los vectores de los resultados en una lista y retornamos ese objeto
  list_result<-list(aucs,Cs,sigmas,aucs_train,tprs,accs,nsv)
  return(list_result)
}
```

7.3. Función *califica* (Segmento 1)

```
##############################################################################
# CODIGO QUE ILUSTRA COMO CALIFICAR EL SEGMENTO 1 GRAVITY A PARTIR
# DEL MODELO SVM

# PAQUETES REQUERIDOS
# ggplot2: Para visualizar la curva de PD como funci??n del score.
# pROC:    Para calcular el Area under ROC curve (AUC).
#          *Nota: Gini = 2*AUC - 100, si el AUC se expresa entre 0 y 100
##############################################################################


califica<-function(supports,subbase,testfull)
{

  #INPUT:

  #supports: Base que contiene los vectores de soporte (V.S.), con sus respectivas alphas.
  #          Las alphas en esta base ya contienen el signo de la clase, es decir,
  #          representan alpha_i*y_i, i=1,...,#V.S.
  #subbase:  Base de entrenamiento de donde se obtuvo el modelo y de donde surgen los V.S.
  #          Se necesita para obtener las medias y desviaciones para estandarizar
  #          las nuevas observaciones.
  #testfull: Base de prueba, que contiene las observaciones a calificar.

  #OUTPUT:

  #baseout:  Base de prueba recibida como input (testfull), agregando las columnas
  #          correspondientes al score y a la PD. (score_svm y pd_svm).

  ######################
  # VECTORES DE SOPORTE
  ######################

  #ELIMINAMOS LAS COLUMNAS QUE NO NECESITAMOS
  supports_vars<-supports
  supports_vars$X<-NULL
  supports_vars$inc<-NULL
  supports_vars$llavepu<-NULL
  supports_vars$preds<-NULL

  #GUARDAMOS LAS ALPHAS EN OTRO OBJETO
  alphas<-supports_vars$alpha
  supports_vars$alpha<-NULL
```

```
#####################################################################
# BASE DE ENTRENAMIENTO (DE AQUI SURGEN LOS VECTORES DE SOPORTE)
#####################################################################

#ELIMINAMOS LAS COLUMNAS QUE NO NECESITAMOS
basetrain<-subbase
basetrain$X<-NULL
basetrain$inc<-NULL
basetrain$llavepu<-NULL

#OBTENEMOS LAS MEDIAS Y DESVIACIONES DE CADA VARIABLE Y LAS GUARDAMOS EN OBJETOS
basetrain<-as.matrix(basetrain)
basetrain_scaled<-scale(basetrain,center=T,scale=T)
medias<-attr(basetrain_scaled,"scaled:center")
sds<-attr(basetrain_scaled,"scaled:scale")

#ESTANDARIZAMOS LOS VECTORES DE SOPORTE DE ACUERDO A LAS MEDIAS Y DESVIACIONES OBTENIDAS
supports_scaled<-as.matrix(supports_vars)
supports_scaled<-scale(supports_scaled,center=medias,scale=sds)

#################
# BASE DE PRUEBA
#################

#ELIMINAMOS LAS COLUMNAS QUE NO NECESITAMOS
basetest<-testfull
basetest$X<-NULL
basetest$preds<-NULL

#GUARDAMOS LA MARCA DE INCUMPLIMIENTO Y LAS LLAVES EN OBJETOS
inc_test<-basetest$inc
basetest$inc<-NULL
llaves_test<-basetest$llavepu
basetest$llavepu<-NULL

#ESTANDARIZAMOS LA BASE DE PRUEBA DE ACUERDO A LAS MEDIAS Y DESVIACIONES OBTENIDAS
basetest_scaled<-as.matrix(basetest)
basetest_scaled<-scale(basetest_scaled,center=medias,scale=sds)

########################################
# FUNCION PARA CALCULAR EL SCORE Y LA PD
########################################

scores<-function(test_sc,supports_sc,alphas,intercept,A,B)
{
```

```
  sigma<-0.1 #Parametro del kernel radial
  kerexp<-rep(0,nrow(supports_sc))
  f<-rep(0,nrow(test_sc))
  prob<-rep(0,nrow(test_sc))

  for (j in 1:nrow(test_sc)){
    for (i in 1:nrow(supports_sc)){
      aux<-norm(as.matrix(test_sc[j,])-as.matrix(supports_sc[i,]),"F")
      #Norma-2 de la diferencia de observaciones
      aux<-aux*aux #Norma-2 al cuadrado
      kerexp[i]<-exp((-sigma)*(aux)) #Exponencial elevado a (-sigma)*(norma al cuadrado)
    }
    f_newobs<-sum(alphas*kerexp) #Suma sobre los vectores de soporte
    f_newobs<-f_newobs-intercept
    prob_newobs<-1/(1+exp(A*f_newobs+B))
    f[j]<-f_newobs
    prob[j]<-prob_newobs
  }
  out<-list(f,prob)
}

#PARAMETROS PARA EL SEGMENTO 1
intercept<-(-0.09309717) #Intercepto del score
A<-(-1.744097) #Coeficiente de la curva de PD
B<-(0.02370129) #Intercepto de la curva de PD

#Ejecutamos la funcion -scores- para calificar la base de prueba
out<-scores(basetest_scaled,supports_scaled,alphas,intercept,A,B)

#Guardamos las salidas en objetos
f<-out[[1]] #score_svm
prob<-out[[2]] #pd_svm

#AGREGAMOS LAS LLAVES, LAS MARCAS DE INC., EL SCORE Y LA PD A LA BASE DE PRUEBA
basetest$llavepu<-llaves_test
basetest$inc<-inc_test
basetest$score_svm<-f
basetest$pd_svm<-prob

#Base calificada que va a retornar la funcion
baseout<-basetest
}

################################################################################
################################################################################
```

```
#Tomamos 2000 observaciones seudoaleatoriamente de la base de prueba, como ejemplo
set.seed(45763)
seg1_prueba<-seg1test_full[sample(nrow(seg1test_full), 2000),]

#Numero de malos en la muestra
sum(seg1_prueba$inc==1)
#Numero de buenos en la muestra
sum(seg1_prueba$inc==0)

#Llamamos a la funcion -califica- para calificar esta muestra
seg1_califica_prueba<-califica(seg1_supports,seg1_subbase,seg1_prueba)

#Visualizamos la curva de PD para esta muestra
#Los malos se concentran en los scores (y pd's) mas altos

library(ggplot2)

qplot(seg1_califica_prueba$score_svm,seg1_califica_prueba$pd_svm,
      color=factor(seg1_califica_prueba$inc))+geom_point(size=3)+
  xlab("Score SVM")+ylab("PD SVM")

#####################################################################
# CALCULANDO AREA UNDER ROC CURVE (AUC)
#####################################################################

#AUC sobre la muestra tomada
#El ordenamiento generado por (score_svm) y por (pd_svm) es el mismo

library(pROC)

rocobj<-plot.roc(seg1_califica_prueba$inc, seg1_califica_prueba$score_svm,
        percent=TRUE, ci=TRUE,print.auc=TRUE)
rocobj$auc
#88.65

rocobj<-plot.roc(seg1_califica_prueba$inc, seg1_califica_prueba$pd_svm,
        percent=TRUE, ci=TRUE,print.auc=TRUE)
rocobj$auc
#88.65
```

7.4. Función *califica* (Segmento 4)

```
##############################################################################
# CODIGO QUE ILUSTRA COMO CALIFICAR EL SEGMENTO 4 GRAVITY A PARTIR
# DEL MODELO SVM

# PAQUETES REQUERIDOS
# ggplot2: Para visualizar la curva de PD como función del score.
# pROC:    Para calcular el Area under ROC curve (AUC).
#          *Nota: Gini = 2*AUC - 100, si el AUC se expresa entre 0 y 100
##############################################################################

califica<-function(supports,subbase,testfull)
{

  #INPUT:

  #supports: Base que contiene los vectores de soporte (V.S.), con sus respectivas alphas.
  #          Las alphas en esta base ya contienen el signo de la clase, es decir,
  #          representan alpha_i*y_i, i=1,...,#V.S.
  #subbase:  Base de entrenamiento de donde se obtuvo el modelo y de donde surgen los V.S.
  #          Se necesita para obtener las medias y desviaciones para estandarizar
  #          las nuevas observaciones.
  #testfull: Base de prueba, que contiene las observaciones a calificar.

  #OUTPUT:

  #baseout:  Base de prueba recibida como input (testfull), agregando las columnas
  #          correspondientes al score y a la PD. (score_svm y pd_svm).

######################
# VECTORES DE SOPORTE
######################

#ELIMINAMOS LAS COLUMNAS QUE NO NECESITAMOS
supports_vars<-supports
supports_vars$X<-NULL
supports_vars$inc<-NULL
supports_vars$llavepu<-NULL
supports_vars$preds<-NULL

#GUARDAMOS LAS ALPHAS EN OTRO OBJETO
alphas<-supports_vars$alpha
supports_vars$alpha<-NULL
```

```
#####################################################################
# BASE DE ENTRENAMIENTO (DE AQUI SURGEN LOS VECTORES DE SOPORTE)
#####################################################################

#ELIMINAMOS LAS COLUMNAS QUE NO NECESITAMOS
basetrain<-subbase
basetrain$X<-NULL
basetrain$inc<-NULL
basetrain$llavepu<-NULL

#OBTENEMOS LAS MEDIAS Y DESVIACIONES DE CADA VARIABLE Y LAS GUARDAMOS EN OBJETOS
basetrain<-as.matrix(basetrain)
basetrain_scaled<-scale(basetrain,center=T,scale=T)
medias<-attr(basetrain_scaled,"scaled:center")
sds<-attr(basetrain_scaled,"scaled:scale")

#ESTANDARIZAMOS LOS VECTORES DE SOPORTE DE ACUERDO A LAS MEDIAS Y DESVIACIONES OBTENIDAS
supports_scaled<-as.matrix(supports_vars)
supports_scaled<-scale(supports_scaled,center=medias,scale=sds)

#################
# BASE DE PRUEBA
#################

#ELIMINAMOS LAS COLUMNAS QUE NO NECESITAMOS
basetest<-testfull
basetest$X<-NULL
basetest$preds<-NULL

#GUARDAMOS LA MARCA DE INCUMPLIMIENTO Y LAS LLAVES EN OBJETOS
inc_test<-basetest$inc
basetest$inc<-NULL
llaves_test<-basetest$llavepu
basetest$llavepu<-NULL

#ESTANDARIZAMOS LA BASE DE PRUEBA DE ACUERDO A LAS MEDIAS Y DESVIACIONES OBTENIDAS
basetest_scaled<-as.matrix(basetest)
basetest_scaled<-scale(basetest_scaled,center=medias,scale=sds)


########################################
# FUNCION PARA CALCULAR EL SCORE Y LA PD
########################################

scores<-function(test_sc,supports_sc,alphas,intercept,A,B)
{
```

```
  sigma<-0.1 #Parametro del kernel radial
  kerexp<-rep(0,nrow(supports_sc))
  f<-rep(0,nrow(test_sc))
  prob<-rep(0,nrow(test_sc))

  for (j in 1:nrow(test_sc)){
    for (i in 1:nrow(supports_sc)){
      aux<-norm(as.matrix(test_sc[j,])-as.matrix(supports_sc[i,]),"F")
      #Norma-2 de la diferencia de observaciones
      aux<-aux*aux #Norma-2 al cuadrado
      kerexp[i]<-exp((-sigma)*(aux)) #Exponencial elevado a (-sigma)*(norma al cuadrado)
    }
    f_newobs<-sum(alphas*kerexp) #Suma sobre los vectores de soporte
    f_newobs<-f_newobs-intercept
    prob_newobs<-1/(1+exp(A*f_newobs+B))
    f[j]<-f_newobs
    prob[j]<-prob_newobs
  }
  out<-list(f,prob)
}

#PARAMETROS PARA EL SEGMENTO 4
intercept<-(-0.1106492) #Intercepto del score
A<-(-1.518724) #Coeficiente de la curva de PD
B<-(-0.09612522) #Intercepto de la curva de PD

#Ejecutamos la funcion -scores- para calificar la base de prueba
out<-scores(basetest_scaled,supports_scaled,alphas,intercept,A,B)

#Guardamos las salidas en objetos
f<-out[[1]] #score_svm
prob<-out[[2]] #pd_svm

#AGREGAMOS LAS LLAVES, LAS MARCAS DE INC., EL SCORE Y LA PD A LA BASE DE PRUEBA
basetest$llavepu<-llaves_test
basetest$inc<-inc_test
basetest$score_svm<-f
basetest$pd_svm<-prob

#Base calificada que va a retornar la funcion
baseout<-basetest
}

################################################################################
################################################################################
```

```
#Tomamos 2000 observaciones seudoaleatoriamente de la base de prueba, como ejemplo
set.seed(45763)
seg4_prueba<-seg4test_full[sample(nrow(seg4test_full), 2000),]

#Numero de malos en la muestra
sum(seg4_prueba$inc==1)
#Numero de buenos en la muestra
sum(seg4_prueba$inc==0)

#Llamamos a la funcion -califica- para calificar esta muestra
seg4_califica_prueba<-califica(seg4_supports,seg4_subbase,seg4_prueba)

#Visualizamos la curva de PD para esta muestra
#Los malos se concentran en los scores (y pd's) mas altos

library(ggplot2)

qplot(seg4_califica_prueba$score_svm,seg4_califica_prueba$pd_svm,
      color=factor(seg4_califica_prueba$inc))+geom_point(size=3)+
  xlab("Score SVM")+ylab("PD SVM")

#####################################################################
# CALCULANDO AREA UNDER ROC CURVE (AUC)
#####################################################################

#AUC sobre la muestra tomada
#El ordenamiento generado por (score_svm) y por (pd_svm) es el mismo

library(pROC)

rocobj<-plot.roc(seg4_califica_prueba$inc, seg4_califica_prueba$score_svm,
        percent=TRUE, ci=TRUE,print.auc=TRUE)
rocobj$auc
#83.41

rocobj<-plot.roc(seg4_califica_prueba$inc, seg4_califica_prueba$pd_svm,
        percent=TRUE, ci=TRUE,print.auc=TRUE)
rocobj$auc
#83.41
```

7.5. Regresiones logísticas (Segmentos 1 y 4)

```
##############################################################################
# REGRESIONES LOGISTICAS: SEGMENTO 1 Y 4
# Se ajustan regresiones logisticas para comparar el desempeño con los
# modelos obtenidos mediante SVM
##############################################################################


#####################################################################
# SEGMENTO 1
#####################################################################

seg1_train<-desarrollo_seg1_train
seg1_train$llavepu<-NULL
seg1_train$cosecha<-NULL
seg1_train$MODELO<-NULL
seg1_train$train<-NULL

seg1_train$Re_Antig_Empleo<-NULL
seg1_train$Re_Tasa_Incidencias_12<-NULL
seg1_train$Re_Num_Cuentas_Ab<-NULL
seg1_train$Re_Tipo_Contrato_Lab<-NULL

seg1_reglog <- glm(Y2 ~ .,family = binomial, data = seg1_train)

# INFORMACION SOBRE EL MODELO (BETAS, P-VALUES, ETC.)
summary(seg1_reglog)

########################
# BASE DE ENTRENAMIENTO
########################

pd_reglog<-predict(seg1_reglog,newdata=seg1_train,type="response")

seg1_train$pd_reglog<-pd_reglog

roccurve<-plot.roc(seg1_train$Y2, seg1_train$pd_reglog,
                   percent=TRUE, ci=TRUE,print.auc=TRUE)
roccurve$auc
#84.05

########################
# BASE DE PRUEBA
########################
```

```
seg1_test<-desarrollo_seg1_test
seg1_test$llavepu<-NULL
seg1_test$cosecha<-NULL
seg1_test$MODELO<-NULL
seg1_test$train<-NULL

seg1_test$Re_Antig_Empleo<-NULL
seg1_test$Re_Tasa_Incidencias_12<-NULL
seg1_test$Re_Num_Cuentas_Ab<-NULL
seg1_test$Re_Tipo_Contrato_Lab<-NULL

pd_reglog<-predict(seg1_reglog,newdata=seg1_test,type="response")

seg1_test$pd_reglog<-pd_reglog

roccurve<-plot.roc(seg1_test$Y2, seg1_test$pd_reglog,
                   percent=TRUE, ci=TRUE,print.auc=TRUE)
roccurve$auc
#80.89


######################################################################
# SEGMENTO 4
######################################################################

seg4_train<-desarrollo_seg4_train
seg4_train$llavepu<-NULL
seg4_train$cosecha<-NULL
seg4_train$MODELO<-NULL
seg4_train$train<-NULL

seg4_reglog <- glm(Y2 ~ .,family = binomial, data = seg4_train)

# INFORMACION SOBRE EL MODELO (BETAS, P-VALUES, ETC.)
summary(seg4_reglog)

########################
# BASE DE ENTRENAMIENTO
########################

pd_reglog<-predict(seg4_reglog,newdata=seg4_train,type="response")

seg4_train$pd_reglog<-pd_reglog
```

```
roccurve<-plot.roc(seg4_train$Y2, seg4_train$pd_reglog,
                   percent=TRUE, ci=TRUE,print.auc=TRUE)
roccurve$auc
#77.93

########################
# BASE DE PRUEBA
########################

seg4_test<-desarrollo_seg4_test
seg4_test$llavepu<-NULL
seg4_test$cosecha<-NULL
seg4_test$MODELO<-NULL
seg4_test$train<-NULL

pd_reglog<-predict(seg4_reglog,newdata=seg4_test,type="response")

seg4_test$pd_reglog<-pd_reglog

roccurve<-plot.roc(seg4_test$Y2, seg4_test$pd_reglog,
                   percent=TRUE, ci=TRUE,print.auc=TRUE)
roccurve$auc
#78.61
```

Referencias

[1] Bishop, C. (2010), *Pattern Recognition and Machine Learning*. Springer.

[2] Hastie, T. *et al.* (2008), *The elements of Statistical Learning*. Springer.

[3] Hastie, T. *et al.* (2010), *Generalized Linear Models via Coordinate Descent*. Journal of Statistical Sofware.

[4] Karatzoglou, A. *et al.* (2013), *kernlab: An S4 Package for Kernel Methods in R*. The Comprehensive R Archive Network.

[5] Shiloh, R. (2007), *Support Vector Machines for Classification and Regression*. M.Sc. Thesis, McGill University.

[6] Rifkin, R. (2002), *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. Ph. D. Thesis, MIT.