Trabajo Fin de Máster

Modelos de Regresión con Alta Dimensión en el Número de Covariables

Iván Dopazo Iglesias

Máster en Técnicas Estadísticas

Curso 2015-2016

Propuesta de Trabajo Fin de Máster

Título en galego: Modelos de Regresión con Alta Dimensión no Número de Covariables
Título en español: Modelos de Regresión con Alta Dimensión en el Número de Covariables
English title: Regression Models with High Dimension in the Number of Covariates
Modalidad: Modalidad A
Autor/a: Iván Dopazo Iglesias, Universidad de Santiago de Compostela
Director/a: Wenceslao González Manteiga, Universidad de Santiago de Compostela
Breve resumen del trabajo: Debido al gran aumento que sufrimos en los últimos años en la cantidad de datos de los que disponemos, necesitamos nuevas técnicas estadísticas que nos permitan obtener información de ellos. Para eso, la aplicación de métodos que puedan operar cuando el número de covaribles (p) es mayor que el de individuos (N) es imprescindible. Por eso, en este trabajo explicaremos estas nuevas metodologías y las aplicaremos a dos situaciones distintas en la variable respuesta: cuando es continua y cuando es binaria. Estas metodologías se van a basar en intentar reducir el número de variables que intervienen en los modelos (métodos d selección de subconjuntos), reducir el número de variables mediante penalizaciones en la estimación (métodos de regularización) y reducir el número de variables mediante la estimación de combinaciones lineales de las mismas (modelos de reducción de la dimensión). Finalmente, realizaremos una comparación de todos los modelos, incluyendo en esta comparación los modelo GAM.

Índice

Índice de gráficos

Índice de tablas

Introducción

En la actualidad, disponemos de una gran cantidad de datos, desde grandes bases de datos biológicos hasta datos económicos y del comportamiento del ser humano. La generalización de la utilización de páginas web, redes sociales, dispositivos móviles, etc, hace que la cantidad de datos que se recogen y almacenan no deje de crecer, por lo que cada vez es más compleja la tarea de analizarlos. Para poder analizar esta enorme cantidad de datos, los métodos estadísticos que se utilizaban dejan de ser útiles por lo que es necesario el desarrollo de otros nuevos para adaptarse así a los nuevos tiempos.

El objetivo de este trabajo es exponer algunos métodos que nos permitan llevar a cabo el análisis de estas grandes cantidades de datos, centrándonos en el caso de que el número de variables que incorporemos al modelo (p) sea mayor que el número de casos (n).

En el capítulo 1, haremos un breve repaso de los modelos de regresión convencionales, desde el modelo lineal simple hasta los modelos aditivos. En el capítulo 2, expondremos los métodos más utilizados para el caso en el que $p > n$, incluyendo métodos de selección de subconjuntos, métodos de regularización, modelos de reducción de la dimensión y modelos aditivos generalizados, esta vez centrándonos en la situación de alta dimensión. En el capítulo 3, explicaremos brevemente algunas de las funciones implementadas en el lenguaje de programación R por distintos autores para los modelos anteriormente expuestos. Y, por útlimo, en el capítulo 4, aplicaremos estas funciones a una base de datos real pero en dos situaciones distintas: en una, la variable respuesta de nuestro modelo será continua (modelos de regresión); en cambio, en la segunda situación, la variable respuesta será binaria (modelos de clasificación). Por eso, a lo largo de todo el trabajo, iremos ampliando algunos de los modelos para la situación específica de regresión logística.

1. Modelos de regresión

Los modelos de regresión intentan explicar la relación existente entre variables. Además de conocer su relación, se pueden utilizar estos modelos para predecir los valores que tendrán ciertas variables (variables dependientes) en función de los valores de otras de variables (variables independientes). Estos modelos pueden ser de dos tipos: modelos deterministas, en este caso, conociendo los valores de las variables independientes, se puede conocer con exactitud el valor de las variables dependientes; o modelos estocásticos, los cuales tienen incorporados una componente aleatoria impredecible, bien causada por el error de medida de las variables, por la influencia de variables no controlables o por una aleatoriedad intrínseca a la variable respuesta. En nuestro caso, estudiaremos los modelos de regresión estocásticos.

1.1. Modelo de regresión lineal simple

En esta sección, presentaremos los modelos de regresión lineales simples y su estimación [9]. En términos generales, siendo (X, Y) un m.a.s., la función de regresión es de la forma

$$m(x) = \mathbb{E}(Y|X = x) \tag{1.1}$$

para cada posible valor x de X. Entonces, podemos descomponer la variable respuesta como

$$Y = m(X) + \varepsilon, \tag{1.2}$$

donde ε es el error que debe cumplir que $\mathbb{E}(\varepsilon|X = x) = 0$ para todo x. De momento supondremos que tanto X como Y son univariantes.

1.1.1. Hipótesis del modelo

Las hipótesis básicas de este modelo son:

- Linealidad. La función de regresión es una linea recta:

$$Y = \beta_0 + \beta_1 X + \varepsilon, \tag{1.3}$$

  donde β_0 y β_1 son parámetro desconocidos y ε es una variable aleatoria no observable que continene la variabilidad de la variable respuesta que no se puede atribuir a la variable explicativa.
- Homocedasticidad. La varianza del error es constante, es decir, es la misma cualquiera que sea el valor de la variable explicativa:

$$Var(\varepsilon|X = x) = \sigma^2 \tag{1.4}$$

- Normalidad. El error sigue una distribución normal:

$$\varepsilon \sim N(0, \sigma^2) \tag{1.5}$$

- Independencia. Las variables aleatorias $\varepsilon_1, ..., \varepsilon_2$, que representan los errores, son independientes entre ellas.

1.1.2. Tipos de diseño

Para poder realizar la estimación de los parámetros β_0 y β_1 necesitamos una muestra, la cual puede ser de dos tipos:

- Diseño fijo. Los valores de la varible explicativa están fijados previamente por el experimentador. En este caso, solamente el error y, en consecuencia, la variable respuesta, son aleatorios. La muestra de un diseño fijo sería:

$$(x_1, Y_1), ..., (x_n, Y_n) \tag{1.6}$$

- Diseño aleatorio. En este caso ambas variables son aleatorias. La muestra resultante sería:

$$(X_1, Y_1), ..., (X_n, Y_n) \tag{1.7}$$

1.1.3. Estimación de los parámetros por mínimos cuadrados

Una vez estimados los parámetros $\hat{\beta}_0$ y $\hat{\beta}_1$, daríamos una predicción de la variable dependiente Y a partir del valor x de la variable independiente de la forma $\hat{\beta}_0 + \hat{\beta}_1 x$. Aplicando este método a los datos muestrales, para el valor observado x_i, tendríamos la predicción $\hat{\beta}_0 + \hat{\beta}_1 x_i$, mientras que hemos observado Y_i. Por eso, los errores de predicción serían

$$\hat{\varepsilon}_i = Y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i \qquad \text{para } i \in \{1, ..., n\} \tag{1.8}$$

que se conocen como residuos de la regresión.

El método de mínimos cuadrados consiste en escoger los $\hat{\beta}_0$ y $\hat{\beta}_1$ que minimicen esos residuos. Para realizar esto, se minimiza la suma de los residuos al cuadrado. Se ponen al cuadrado para evitar que los residuos negativos se compensen con los positivos. Así, los estimadores $\hat{\beta}_0$ y $\hat{\beta}_1$ por mínimos cuadrados son tales que

$$\sum_{i=1}^{N} \left(Y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i \right)^2 = \min_{\beta_0, \beta_1} \sum_{i=1}^{N} \left(Y_i - \beta_0 - \beta_1 x_i \right)^2 . \tag{1.9}$$

Como resultado se obtienen los estimadores

$$\hat{\beta}_0 = \overline{Y} - \frac{S_{xY}}{S_x^2} \overline{x} \qquad \hat{\beta}_1 = \frac{S_{xY}}{S_x^2}, \tag{1.10}$$

donde $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ son las medias de la variable explicativa y la variable respuesta, respectivamente, $S_{xY} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(Y_i - \overline{Y})$ es la covarianza entre la variable dependiente y la independiente, y $S_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$ es la varianza de la variable explicativa.

Es decir, la recta de regresión estimada por mínimos cuadrados es aquella que pasa por el vector de medias $(\overline{x}, \overline{Y})$ con pendiente $\hat{\beta}_1 = \frac{S_{xY}}{S_x^2}$. La varianza del error estimada es

$$\hat{\sigma}^2 = \frac{1}{n-2}\sum_{i=1}^{n}\hat{\varepsilon}_i^2 = \frac{1}{n-2}\sum_{i=1}^{n}\left(Y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right)^2. \tag{1.11}$$

Es la suma de cuadrados de los residuos pero, en este caso, dividido por $n-2$ para que el estimador sea insesgado.

1.2. Modelo de regresión lineal general

1.2.1. Formulación del modelo

El modelo de regresión lineal múltiple es una extensión del modelo de regresión lineal [9]. Es decir, en este caso, X contiene más de una covariable. Así, este modelo aplicado a una muestra de diseño fijo se puede expresar como

$$Y_i = \beta_0 + \beta_1 x_{i,1} + ... + \beta_{p-1} x_{i,p-1} + \varepsilon_i, \tag{1.12}$$

siendo Y_i la variable respuesta del i-ésimo individio, $x_{i,1}, ..., x_{i,p-1}$ las variables explicativas del mismo individuo y ε_i el error asociado a dicho individuo. En este caso también se asume que ε_i satisface las hipótesis de homocedasticidad, normalidad e independencia.

En notación matricial, el modelo de regresión lineal múltiple quedaría

$$\begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1,p-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{n,p-1} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_{p-1} \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix} \tag{1.13}$$

Cuya expresión abreviada es

$$Y = X\beta + \varepsilon, \tag{1.14}$$

donde Y es el vector de respuestas, X una matriz n x p, donde cada fila representa a un individuo y cada columna a una covariable, siendo la primera columna de unos para incluir al intercepto; β es el vector de parámetros y ε el vector de los errores que verifica $\varepsilon \sim N_n(0, \sigma^2 I_n)$, siendo σ^2 la varianza del error y I_n la matriz identidad de orden n.

1.2.2. Estimación de los parámetros

En esta sección estudiaremos el problema de la estimación de los parámetro β y σ^2. Igual que en el caso de el modelo de regresión simple, estimaremos β por mínimos cuadrados.

1.2.2.1. Estimación de β

Escogeremos el estimador $\hat{\beta}$ que satisfaga

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - x_i\beta)^2, \tag{1.15}$$

siendo x_i la fila i-ésima de la matriz X. En notación matricial,

$$\min_{\beta} (Y - X\beta)'(Y - X\beta). \tag{1.16}$$

Resolviendo este problema de optimización, obtenemos el estimador

$$\hat{\beta} = (X'X)^{-1}X'Y. \tag{1.17}$$

Para que este estimador esté bien definido, la matriz $X'X$ tiene que ser no singular, es decir, debe tener inversa. Esta matriz es cuadrada de orden p, simétrica y semidefinida positiva. Su rango coincide con la dimensión del espacio lineal en el que se encuentran los vectores x_i. Así, si los vectores son linealmente independientes, harían falta al menos p individuos ($n \geq p$) para que $X'X$ sea no singular.

Una vez obtenidos los estimadores de los parámetros $\hat{\beta}$ se pueden calcular las predicciones para los individuos de la muestra de la siguiente forma:

$$\hat{Y}_i = x_i\hat{\beta} \quad i \in \{1, .., n\}, \quad \text{o equivalentemente,} \quad \hat{Y} = X\hat{\beta} \tag{1.18}$$

1.2.2.2. Estimación de la varianza

Como en la regresión simple, los residuos en el modelo lineal general se definen de la siguiente forma:

$$\hat{\varepsilon}_i = Y_i - \hat{Y}_i = Y_i - x_i\hat{\beta} \qquad i \in \{1, ..., n\}. \tag{1.19}$$

También se puede formar un vector de residuos

$$Y - \hat{Y} = (I_n - H)Y = MY, \tag{1.20}$$

con $M = I_n - H$, lo que se llama matriz generadora de residuos. M es una matriz simétrica, idempotente y de rango $(n - p)$.

Como los errores no se observan, estimaremos su varianza mediante los residuos. El estimador de la varianza del error sería

$$\hat{\sigma}^2 = \frac{1}{n-p}\sum_{i=1}^{n}\hat{\varepsilon}_i^2 = \frac{1}{n-p}\sum_{i=1}^{n}(Y_i - x_i\hat{\beta})^2 = \frac{RSS}{n-p}, \tag{1.21}$$

donde RSS es la suma de los residuos al cuadrado y, en el denominador, ponemos $(n-p)$, a diferencia de $(n-2)$ que poníamos en el modelo lineal simple, ya que en ése solamente estimábamos dos parámetros, mientras que ahora estimamos p parámetros.

Entonces $RSS = \hat{\varepsilon}'\hat{\varepsilon}$, donde $\hat{\varepsilon} = (I_n - H)Y = MY$. Por eso, podemos escribir

$$RSS = \hat{\varepsilon}'\hat{\varepsilon} = (MY)'MY = Y'M'MY \overset{(a)}{=} Y'MY, \tag{1.22}$$

donde la igualdad (a) se deduce por ser M una matriz simétrica e idempotente.

1.3. Modelo de regresión logística

1.3.1. Formulación del modelo

En los dos modelos anteriores, la variable respuesta Y era continua pero, en este caso, es binaria, es decir, solamente toma los valores 0 y 1, identificándolos como fracaso y éxito, respectivamente. El modelo logístico consiste en

$$Y_i = \pi(x_i) + \varepsilon_i$$
$$\pi(x) = P(Y = 1|X = x), \tag{1.23}$$

donde ε_i es una variable aleatoria i.i.d. con media 0 y varianza σ^2 y $\pi(x)$ es la probabilidad de éxito condicionada a cada valor de la variable explicativa. Al ser una probabilidad, $\pi(x)$ está contenida en el intervalo [0,1], por lo que para conseguir que se encuentre en el intervalo $(-\infty, +\infty)$ debemos aplicarle una función link, que en nuestro caso será la función logit:

$$g(p) = \log\frac{p}{1-p} \qquad \forall p \in [0,1]. \tag{1.24}$$

Como p va a ser sustituida por la probabilidad de éxito, la función logística consiste en aplicar un logaritmo (en base e) al cociente entre la probabilidad de éxito, p, y la probabilidad de fracaso, $(1-p)$. Este cociente es conocido como odds:

$$Odds(Y) = \frac{P(Y=1)}{P(Y=0)}. \tag{1.25}$$

Al ser un cociente de probabilidades, la Odds puede tomar valores en el intervalo [0,+∞]. Para que tome valores en el intervalo $[-\infty, +\infty]$, debemos aplicarle un logaritmo, por lo que resultará

$$\log\frac{\pi(x,\beta)}{1-\pi(x,\beta)} = x'\beta. \tag{1.26}$$

Para poder expresar el modelo en función de la probabilidad de éxito, debemos utilizar la inversa de la función logit:

$$g^{-1}(x) = \frac{e^x}{1+e^x}, \tag{1.27}$$

por lo que, finalmente, el modelo logístico consiste en expresar la probabilidad de éxito de la siguiente manera:

$$\pi(x,\beta) = g^{-1}(x'\beta) = \frac{e^{x'\beta}}{1+e^{x'\beta}}. \tag{1.28}$$

La odds se interpreta en términos relativos, es decir, es la probabilidad de pertenecer al grupo de éxito en relación a la probabilidad de pertenecer al grupo de fracaso. Que la odds sea mayor que uno significa que hay mayor probabilidad de pertenecer al grupo de éxito que al de fracaso. En cambio, si la odds es menor que 1, implica que la probabilidad de pertenecer al grupo de éxito es menor que al de fracaso.

Otro forma de interpretarlo es mediante la odds ratio. La odds ratio entre dos poblaciones respecto a una variable binaria se define como

$$OddsRatio = \frac{Odds(Y|Pob.2)}{Odds(Y|Pob.1)} = \frac{P(Y=1|Pob.2)/P(Y=0|Pob.2)}{P(Y=1|Pob.1)/P(Y=0|Pob.1)}. \tag{1.29}$$

Esta cantidad es por la que se multiplica la odds al pasar de la población 1 a la 2. Para el caso de $K > 2$ poblaciones, el modelo tiene la forma

$$\begin{aligned}
\log\frac{P(G=1|X=x)}{P(G=K|X=x)} &= \beta_{10} + \beta_1^T x \\
\log\frac{P(G=2|X=x)}{P(G=K|X=x)} &= \beta_{20} + \beta_2^T x \\
&\vdots \\
\log\frac{P(G=K-1|X=x)}{P(G=K|X=x)} &= \beta_{(K-1)0} + \beta_{K-1}^T x
\end{aligned} \tag{1.30}$$

es decir,

$$P(G=k|x) = \frac{e^{\beta_{0k}+\beta_k^T x}}{\sum_{i=1}^K e^{\beta_{0i}+\beta_i^T x}}. \tag{1.31}$$

El modelo está especificado en términos de $K-1$ log-odds o transformaciones logit. El modelo utiliza la última clase como denominador de la odds-ratio.

1.3.2. Estimación de los parámetros por máxima verosimilitud

Si tenemos una m.a.s. $(X_1,Y_1),...,(X_n,Y_n)$, $Y_i \sim \text{Bernoulli}(\pi(X_i,\beta))$, la función de verosimilitud adopta la forma:

$$L(\beta) = \prod_{i=1}^n \left[\pi(x_i,\beta)^{y_i}(1-\pi(x_i,\beta))^{1-y_i}\right], \tag{1.32}$$

cuyo logaritmo es

$$\log L(\beta) = \sum_{i=1}^{n} \left[y_i \log \pi(x_i, \beta) + (1 - y_i) \log(1 - \pi(x_i, \beta)) \right]. \tag{1.33}$$

La derivada parcial respecto del parámetro β será

$$\frac{\partial \log L(\beta)}{\partial \beta} = \sum_{i=1}^{n} \frac{\partial \pi(x_i, \beta)}{\partial \beta} \frac{1}{\pi(x_i, \beta)(1 - \pi(x_i, \beta))} [y_i - \pi(x_i, \beta)]. \tag{1.34}$$

Sustituyendo una función de regresión cualquiera $\pi(x, \beta)$ por nuestra función

$$\pi(x, \beta) = \frac{e^{x'\beta}}{1 + e^{x'\beta}}, \tag{1.35}$$

obtenemos,

$$\frac{\partial \pi(x, \beta)}{\partial \beta} = x' \pi(x, \beta)(1 - \pi(x, \beta)) \tag{1.36}$$

y

$$\frac{\partial \log L(\beta)}{\partial \beta} = \sum_{i=1}^{n} x_i' [y_i - \pi(x_i, \beta)] = 0. \tag{1.37}$$

Éstas son las ecuaciones de máxima verosimilitud. En este caso de regresión logística, estas ecuaciones no admiten solución explícita ya que $\pi(x, \beta)$ no es función lineal de β. Por eso debemos utilizar métodos iterativos, que en nuestro caso será el método de Newton-Raphson [16]. En primer lugar, debemos calcular la matriz hessiana, cuya expresión es

$$\frac{\partial^2 \log L(\beta)}{\partial \beta^2} = -\sum_{i=1}^{n} x_i x_i' \pi(x_i, \beta)(1 - \pi(x_i, \beta)). \tag{1.38}$$

En el caso multivariante, $x_i x_i'$ es una matriz simétrica, semidefinida positiva y de rango uno, y sigue teniendo las mismas características al multiplicarla por $\pi(x_i, \beta)(1-\pi(x_i, \beta))$, que es mayor que cero y por tanto la suma será semidefinida positiva. Es más, será definida positiva siempre que los vectores x_i no estén contenidos en un espacio lineal de dimensión inferior. En consecuencia, la matriz hessiana será definida negativa y la raíz de las ecuaciones de verosimilitud es un máximo de la función de verosimilitud y, por lo tanto, un estimador de máxima verosimilitud.

Construimos una nueva matriz diagonal con los valores $\pi(x_i, \beta)(1 - \pi(x_i, \beta))$

$$V = \begin{pmatrix} \pi(x_1, \beta)(1 - \pi(x_1, \beta)) & & 0 \\ & \ddots & \\ 0 & & \pi(x_n, \beta)(1 - \pi(x_n, \beta)) \end{pmatrix}$$

Como $\pi(x_i, \beta)$ es la probabilidad de éxito ajustada por el modelo para cada dato muestral, $\pi(x_i, \beta)(1 - \pi(x_i, \beta))$ es la varianza de la variable respuesta ya que $Y_i \sim$ Bernoulli$(\pi(x_i, \beta))$. Entonces, la matriz hessiana se puede escribir

$$\frac{\partial^2 \log L(\beta)}{\partial \beta^2} = -\sum_{i=1}^{n} x_i x_i' \pi(x_i, \beta)(1 - \pi(x_i, \beta)) = -X'VX \tag{1.39}$$

Comenzando con β^{old}, una simple actualización de Newton es

$$\beta^{new} = \beta^{old} - \left(\frac{\partial^2 \log L(\beta)}{\partial \beta^2} \right)^{-1} \frac{\partial \log L(\beta)}{\partial \beta} \tag{1.40}$$

donde las derivadas son evaluadas en β^{old}.

Es conveniente escribir las score y Hesiana en notación matricial. Si y denota el vector de valores y_i, X la matriz N x $(p + 1)$ de valores x_i, el vector p de probabilidades ajustadas con el i-ésimo elemento $\pi(x_i, \beta^{old})$ y W una matriz diagonal NxN de pesos con el i-ésimo elemento diagonal $\pi(x_i, \beta^{old})(1 - \pi(x_i, \beta^{old}))$. Entonces, tenemos

$$\frac{\partial \log L(\beta)}{\partial \beta} = X^T(y - p) \tag{1.41}$$

$$\frac{\partial^2 \log L(\beta)}{\partial \beta \partial \beta^T} = -X^T W X. \tag{1.42}$$

El paso de Newton es

$$\begin{aligned} \beta^{new} &= \beta^{old} + (X^T W X)^{-1} X^T (y - p) \\ &= (X^T W X)^{-1} X^T W (X\beta^{old} + W^{-1}(y - p)) \\ &= (X^T W X)^{-1} X^T W z. \end{aligned}$$

En la segunda y tercera linea hemos reexpresado el paso de Newton como un paso de mínimos cuadrados ponderados, con la respuesta

$$z = X\beta^{old} + W^{-1}(y - p), \tag{1.43}$$

que se conoce como respuesta ajustada. Estas ecuaciones se resuelven de forma iterativa, cambiando p en cada iteración, igual que ocurre con W y z. Este algoritmo se conoce como mínimos cuadrados iterativamente reponderados (IRLS, iteratively reweighted least squares) ya que cada iteración resuelve el problema de mínimos cuadrados ponderados

$$\beta^{new} \leftarrow \arg\min_{\beta} (z - X\beta)^T W (z - X\beta). \tag{1.44}$$

$\beta = 0$ es un buen valor de comienzo. La convergencia nunca está garantizada, aunque normalmente sí se da.

En los siguientes apartados, estudiaremos algunas extensiones del modelo logístico.

1.3.3. Aproximación cuadrática e inferencia

El parámetro estimado por máxima verosimilitud $\hat{\beta}$ satisface una relación de autoconsistencia: son los coeficientes de un ajuste mínimo cuadrado ponderado, donde las respuestas son

$$z_i = x_i^T \hat{\beta} + \frac{(y_i - \hat{p}_i)}{\hat{p}_i(1-\hat{p}_i)} \tag{1.45}$$

y los pesos son w_i, que ambos dependen del propio $\hat{\beta}$. Además de proveer un algoritmo adecuado, esta conexión con el método de mínimos cuadrados tiene más parecidos:

- La PRSS (suma de los residuos ponderados al cuadrado) es el estadístico χ^2

$$\sum_{i=1}^{N} \frac{(y_i - \hat{p}_i)^2}{\hat{p}_i(1-\hat{p}_i)}, \tag{1.46}$$

  una aproximación cuadrática de la deviance.
- La teoría de verosimilitud asintótica dice que si el modelo es correcto $\hat{\beta}$ es consistente.
- El teorema central del límite muestra que la distribución de $\hat{\beta}$ converge a una distribución $N(\beta, (X^T W X)^{-1})$.
- La contrucción del modelo puede ser costosa ya que hay que realizar iteraciones para cada término que incluimos o excluimos. Pero existen, entre otros, dos métodos que ahorran este proceso, que son *Rao score test*, que se utiliza para incluir términos en el modelo, y *Wald test*, que se utiliza para excluir términos del modelo.

1.3.4. Regresión logística no paramétrica

El modelo logístico con una única variable independiente es

$$log \frac{P(Y=1|X=x)}{P(Y=0|X=x)} = f(x), \tag{1.47}$$

lo que implica

$$P(Y=1|X=x) = \frac{e^{f(x)}}{1+e^{f(x)}}. \tag{1.48}$$

Ajustando de forma suavizada $f(x)$, se obtiene un estimador suavizado de la probabilidad condicional $P(Y=1|x)$. Construyendo el criterio de máxima verosimilitud penalizada

$$\begin{aligned} l(f;\lambda) &= \sum_{i=1}^{N} [y_i \log \pi(x_i) + (1-y_i)\log(1-\pi(x_i))] - \frac{1}{2}\lambda \int \{f''(t)\}^2 dt \\ &= \sum_{i=1}^{N} \left[y_i f(x_i) - \log(1+e^{f(x_t)}) \right] - \frac{1}{2}\lambda \int \{f''(t)\}^2 dt, \end{aligned} \tag{1.49}$$

donde $\pi(x) = P(Y = 1|x)$. Se puede representar $f(x) = \sum_{j=1}^{N} N_j(x)\theta_j$. Se calculan la primera y segunda derivadas

$$\frac{\partial l(\theta)}{\partial \theta} = N^T(y - p) - \lambda\Omega\theta \tag{1.50}$$

$$\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^T} = -N^T W N - \lambda\Omega \tag{1.51}$$

donde p es el N-vector con elementos $\pi(x_i)$ y W es la matriz diagonal de pesos $\pi(x_i)(1 - \pi(x_i))$. La primera derivada (1.50) es no linear en θ, por lo que es necesario utilizar un algoritmo iterativo como en el capítulo 1.3.2. Utilizando el algoritmo Newton-Raphson como en (1.40) y (1.43), las ecuaciones actualizadas se pueden escribir como

$$\begin{aligned} \theta^{new} &= (N^T W N + \lambda\omega)^{-1} N^T W (N\theta^{old} + W^{-1}(y - p)) \\ &= (N^T W N + \lambda\omega)^{-1} N^T W z. \end{aligned} \tag{1.52}$$

Expresando esta ecuación en términos de valores ajustados

$$\begin{aligned} f^{new} &= N(N^T W N + \lambda\omega)^{-1} N^T W (f^{old} + W^{-1}(y - p)) \\ &= S_{\lambda,w} z. \end{aligned} \tag{1.53}$$

La ecuación (1.53) intenta reemplazar $S_{\lambda,w}$ por cualquier regresor no paramétrico y obtener familias generales de modelos de regresión logística no paramétrica. Esto es para el caso de x unidimensional. El caso de x multidimensional se estudia en el apartado 1.4.3 con los modelos aditivos generalizados.

1.4. Modelos aditivos

1.4.1. Modelos aditivos generalizados

Los modelos aditivos generalizados son modelos estadísticos flexibles que pueden ser utilizados para identificar y caracterizar efectos de regresión no lineales. Estos modelos tienen la forma

$$E(Y|X_1, X_2, ..., X_p) = \alpha + f_1(X_1) + f_2(X_2) + ... + f_p(X_p). \tag{1.54}$$

Como de costumbre, $X_1, X_2, ..., X_p$ son variables independientes, Y es la variable dependiente y las f_j son funciones suavizadas. En estos modelos ajustamos cada función y obtenemos un algoritmo para estimar conjuntamente las p funciones.

Mientras la forma suavizada de las funciones f_j hace al modelo más flexible, la aditividad permite una fácil interpretación del mismo. Esta flexibilidad permite producir un ajuste suave y al tiempo detectar comportamientos locales. En general, la media condicional $\mu(X)$ de la respuesta Y se relaciona con una función aditiva de los predictores mediante una función link g:

$$g[\mu(X)] = \alpha + f_1(X_1) + ... + f_p(X_p). \tag{1.55}$$

Algunos ejemplos clásicos de función link son:

- $g(\mu) = \mu$. Se utiliza en modelos lineales y aditivos cuando la respuesta es normal.
- $g(\mu) = \text{logit}(\mu)$ o $g(\mu) = \text{probit}(\mu)$. La función probit se utiliza para modelizar probabilidades binomiales. Es la función de distribución acumulada normal inversa.
- $g(\mu) = \log(\mu)$. Se utiliza en modelos log-lineales o log-aditivos cuando la variable respuesta es de Poisson.

Pero no todas las funciones f_j tienen que ser no paramétricas, pueden ser modelos mixtos, es decir, algunas funciones f_j son paramétricas y otras no paramétricas. Cuando se trata de modelos mixtos, el modelo es de la forma

$$g(\mu) = X^*\theta + f_1(X_1) + ... + f_p(X_p), \tag{1.56}$$

siendo X^* las filas de la matriz X con componentes estrictamente paramétricas y θ su vector de parámetros asociado. Pero el aumento en la flexibilidad que tiene este modelo es a costa de dos nuevos problemas: escoger las funciones suavizadoras y cómo deben ser esas funciones.

1.4.1.1. Función suavizadora

Para resolver estos dos problemas, existen distintas alternativas. En primer lugar, para el problema de escoger la función suavizadora nos encontramos, entre otras, las siguientes opciones, las cuales se explican a continuación:

- Basic splines.
- Kernel smoother.
- Smoothing splines.
- Penalized Splines
- Thin plane splines.

Basic splines

Los B-splines de grado l se obtienen fusionando $(l+1)$ polinomios de grado l suavemente en los $(l-1)$ knots interiores. Un B-spline de grado $l = 0$ es de la forma [5]:

$$B_j^0(x) = I_{[k_j, k_{j+1})}(x) = \begin{cases} 1, & k_j \leq x < k_{j+1}; \\ 0, & \text{en otro caso.} \end{cases} \tag{1.57}$$

Los B-splines de orden superior se definen recursivamente como

$$B_j^l(x) = \frac{x - k_j}{k_{j+l} - k_j} B_j^{l-1}(x) + \frac{k_{j+l+1} - x}{k_{j+l+1} - k_{j+1}} B_{j+1}^{l-1}(x). \tag{1.58}$$

Ésto nos lleva a la siguiente clase de funciones: una función $f : [a, b] \to \mathbb{R}$ es un spline polinómico de grado l si satisface las condiciones:

- $f(x)$ es $(l-1)$ veces continuamente diferenciable y
- $f(x)$ es un polinomio de grado l para $x \in [k_j, k_{j+1}), j = 1, ..., m-1$.

El espacio de splines polinómicos es un espacio vectorial $(m+l-1)$-dimensional, y un subespacio del espacio de funciones $(l-1)$ veces continuamente diferenciable. Por tanto, cada spline polinómico puede ser representado por una base de $d = (m+l-1)$ funciones, de la siguiente forma:

$$f(x) = \sum_{j=1}^{d} \beta_j B_j(x). \tag{1.59}$$

Para estimar el modelo de regresión

$$\eta = f(x), \tag{1.60}$$

con f una función (desconocida) suave, debemos construir una base de $(m+l-1)$ B-splines

$$\{B_1(x), ..., B_{m+l-1}(x)\}, \tag{1.61}$$

donde m es el número de knots y l el grado de los polinomios. Evaluamos los B-splines, B_j, en cada valor de la covariable x y ajustamos el siguiente modelo

$$\eta = \sum_{j=1}^{d} \beta_j B_j(x), \tag{1.62}$$

con $d = m + l - 1$.

Según [24], los B-splines tienen dos problemas, la elección del número de knots a utilizar y la posición de los mismos. Un número reducido de knots puede resultar en un espacio de funciones que no es lo suficientemente flexible como para capturar la variabilidad de los datos, en cambio, un número muy alto de knots puede llevar a sobreestimar. También la elección de la posición de los knots puede influir profundamente en la estimación.

Kernel smoother

Para el caso de una sola covariable, este método calcula la media de los valores de la muestra que se encuentran en un intervalo centrado en un punto x_i:

$$\hat{f}(x) = Ave(y_i | x_i \in N_k(x)), \tag{1.63}$$

siendo $N_k(x)$ el conjunto de k puntos más cercanos a x_i en distancia cuadrática, y Ave denota la media. Así, se estima la función suavizadora para cada punto de la muestra. Pero este método resulta en una función discontinua, lo que podemos solucionar simplemente aplicando unos pesos de cero a los valores lejanos de x_i. Para eso, se utiliza la media de Nadaraya-Watson kernel-weighted [23]:

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)}, \tag{1.64}$$

con

$$K_\lambda(x_o, x) = D\left(\frac{|x - x_0|}{h_\lambda(x_0)}\right) \tag{1.65}$$

y D una función de densidad. Esta definición de $K_\lambda(x_o, x)$ es para el caso de ventana adaptativa (tamaño de la ventana variable). Para el caso en que el tamaño de la vetana sea fijo, sustituiríamos el denominador $h_\lambda(x_0)$ por la constante λ y si, además, escogemos la función de densidad

$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{si } |t| \leq 1 \\ \\ 0 & \text{en otro caso,} \end{cases} \tag{1.66}$$

obtendremos el kernel cuadrático de *Epanechnikov*.

Para el caso de más de una covariable, el Nadaraya-Watson kernel smoother ajusta una constante localmente con pesos de un kernel p-dimensional. Sea $b(X)$ un vector de términos polinómicos en X con grado máximo de d, por ejemplo, con d=2 y p=2, obtendremos $b(X) = (1, X_1, X_2, X_1^2, X_2^2, X_1X_2)$. Para cada $x_0 \in \mathbb{R}^p$ resuelve

$$\min_{\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i)(y_i - b(x_i)^T \beta(x_0))^2 \tag{1.67}$$

para producir el ajuste $\hat{f}(x_0) = b(x_0)^T \hat{\beta}(x_0)$. Usualmente el kernel será una función radial, como por ejemplo, el kernel radial Epanechnikov

$$K_\lambda(x_0, x) = D\left(\frac{\|x - x_0\|}{\lambda}\right), \tag{1.68}$$

donde $\|\cdot\|$ es la norma euclídea.

Smoothing splines

La principal ventaja de este método sobre los basic splines es que, en este caso, no tenemos el problema de la selección de los knots ya que utiliza el conjunto máximo de knots. Si consideramos el problema de encontrar de entre todas las funciones $f(x)$ con dos derivadas continuas aquella que minimiza la suma de cuadrados de los residuos penalizada, la solución es [23]

$$RSS(f, \lambda) = \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt, \tag{1.69}$$

donde λ es un parámetro de suavizado fijo. El primer término mide la proximidad a los datos, mientras que el segundo penaliza la curvatura en la función y λ establece el tradeoff entre los dos. Caben destacar dos casos:

- $\lambda = 0$: f puede ser cualquier función que interpole los datos.
- $\lambda = \infty$: es la recta de regresión por mínimos cuadrados.

Se puede demostrar que la ecuación (1.69) tiene solución explícita, finito-dimensional y única, que es un spline cúbico natural con knots en los valores de x_i, $i = 1, ..., N$ [23]. Entonces, tiene N knots y por lo tanto N grados de libertad. Sin embargo, el término de penalización es una penalización en los coeficientes de los spline, los cuales son reducidos hacia el ajuste lineal.

Como la solución es un spline natural, la podemos escribir como

$$f(x) = \sum_{j=1}^{N} N_j(x)\theta_j, \tag{1.70}$$

donde $N_j(x)$ es un conjunto N-dimensional de bases de funciones. Entonces el criterio se reduce a

$$RSS(\theta, \lambda) = (y - N\theta)^T(y - N\theta) + \lambda\theta^T\Omega_N\theta, \tag{1.71}$$

donde $\{N\}_{ij} = N_j(x_i)$ y $\{\Omega_N\}_{jk} = \int N_j''(t)N_k''(t)dt$. La solución al problema es

$$\hat{\theta} = (N^TN + \lambda\Omega_N)^{-1}N^Ty, \tag{1.72}$$

que es una regresión ridge generalizada (capítulo 2.3.1). El ajuste del smoothing spline es

$$\hat{f}(x) = \sum_{j=1}^{N} N_j(x)\hat{\theta}_j. \tag{1.73}$$

Penalized Splines

La metodología de los P-splines, propuesta por Eilers y Marx [13], consiste en utilizar una base para la regresión y modificar la función de verosimilitud introduciendo una penalización basada en diferencias entre coeficientes adyacentes. Este método lo idearon para resolver los dos problemas que surgen en los B-Splines.

Consideremos la regresión de m pares de datos (x_i, y_i) en un conjunto de n B-splines $B_j(\cdot)$. La función mínimo cuadrada objetivo para minimizar es

$$S = \sum_{i=1}^{m} \left\{ y_i - \sum_{j=1}^{n} a_j B_j(x_i) \right\}^2. \tag{1.74}$$

Permitiendo que el número de nodos sea relativamente largo, la curva ajustada mostrará más variación de la necesaria para los datos. Para que el resultado sea menos flexible, algunos autores aplican una penalización en la segunda derivada de la curva estimada y así forman la función objetivo

$$S = \sum_{i=1}^{m} \left\{ y_i - \sum_{j=1}^{n} a_j B_j(x_i) \right\}^2 + \lambda \int_{x_{min}}^{x_{max}} \left\{ \sum_{j=1}^{n} a_j B_j''(x) \right\}^2 dx. \tag{1.75}$$

En cambio, en [13] proponen basar la penalización en diferencias finitas de los coeficientes de los B-splines adyacentes:

$$S = \sum_{i=1}^{m} \left\{ y_i - \sum_{j=1}^{n} a_j B_j(x_i) \right\}^2 + \lambda \sum_{j=k+1}^{n} (\Delta^k a_j)^2. \tag{1.76}$$

Esta aproximación reduce la dimensionalidad del problema a n.

El sistema de ecuaciones que obtenemos al minimizar la ecuación (1.76) se puede escribir como

$$B^T y = (B^T B + \lambda D_k^T D_k) a, \tag{1.77}$$

donde D_k es la representación matricial del operador diferencia Δ^k, y los elementos de B son $b_{ij} = B_j(x_i)$. Cuando $\lambda = 0$, tenemos las ecuaciones normales de la regresión lineal con una base de B-splines y, con $k = 0$, tenemos el caso especial de la regresión ridge (capítulo 2.3.1).

Thin plane splines

Considerando el problema de estimación de una función suavizadora $g(x)$ con n observaciones (y_i, x_i) tal que

$$y_i = g(x_i) + \varepsilon_i, \tag{1.78}$$

donde ε_i es un error aleatorio y x es un vector d-dimensional ($d \leq n$). Thin-plate spline smoothing [39] estima g encontrando la función $\hat{f}$ que minimiza

$$\|y - f\|^2 + \lambda J_{md}(f), \tag{1.79}$$

donde $f = (f(x_1), ..., f(x_n))^T$. $J_{md}(f)$ es una función de penalización que mide la variabilidad de f y λ es el parámetro de suavizado. La penalización de la variabilidad se define como

$$J_{md} = \int ... \int_{\mathbb{R}^d} \sum_{v_1 + ... + v_d = m} \frac{m!}{v_1! ... v_d!} \left(\frac{\partial^m f}{\partial x_1^{v_1} ... \partial x_d^{v_d}} \right)^2 dx_1 ... dx_d. \tag{1.80}$$

Por ejemplo, en caso de suavizado con dos predictores con penalización utilizando la segunda derivada, tenemos

$$J_{22} = \int \int \left(\frac{\partial^2 f}{\partial x_1^2} \right)^2 + \left(\frac{\partial^2 f}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f}{\partial x_2^2} \right)^2 dx_1 dx_2.$$

Si m es escogido tal que $2m > d$, se puede ver que la función para minimizar (1.79) tiene la forma

$$\hat{f}(x) = \sum_{i=1}^{n} \delta_i \eta_{md}(\|x - x_i\|) + \sum_{j=1}^{M} \alpha_j \phi_j(x), \tag{1.81}$$

donde δ y α son coeficientes de vectores que hay que estimar, con δ sujeto a la restricción lineal $T^T \delta = 0$, donde $T_{ij} = \phi_j(x_i)$, y η es la base de funciones utilizada. Las $M = \binom{m+d+1}{d}$ funciones ϕ_i son polinomios linealmente independientes que abarcan el espacio de polinomios en $\mathbb{R}^d$ de grado menor que m.

Definiendo la matriz E como $E_{ij} \equiv \eta_{md}(\|x_i - x_j\|)$, el problema del ajuste del thin plate spline es

$$\begin{aligned} &\text{mín}_{\delta,\alpha} \quad \|y - E\delta - T\alpha\|^2 + \alpha\delta^T E\delta \\ &\text{sujeto a} \quad T^T\delta = 0. \end{aligned} \tag{1.82}$$

Los thin plane splines son muy buenos suavizadores ya que no tenemos que escoger la posición de los knots ni ninguna base de funciones, además de que se pueden utilizar con cualquier número de variables predictoras. El problema de estos suavizadores es el alto coste computacional ya que tiene muchos parámetros desconocidos que debemos estimar. Según [39], el coste computacional cuando hay más de un predictor es proporcional al número de parámetros al cubo.

1.4.1.2. Elección del grado de suavización

Para el problema de como deben ser estas funciones, es decir, el grado de suavización de las mismas, entre otras opciones, se encuentran las siguientes:

a) Unbiased Risk Estimator (UBRE)
b) Cross-Validation
c) Generalized Cross-Validation (GCV)

a) Unbiased Risk Estimator (UBRE)

Este método se utiliza cuando el parámetro de escala es conocido. Una buena forma de estimar el parámetro de suavizado podría ser escoger aquel que haga que $\hat{\mu}$ sea lo más próximo al verdadero $\mu = \mathbb{E}(y)$. Una medida apropiada de esta distancia es M, el error cuadrático medio (MSE) del modelo:

$$\mathbb{E}(M) = \mathbb{E}(\|\mu - X\hat{\beta}\|^2/n) = \mathbb{E}(\|y - Ay\|^2)/n - \sigma^2 + 2tr(A)\sigma^2/n, \tag{1.83}$$

donde la matriz A es una base de funciones.

Entonces, escogemos el parámetro de suavizado que minimiza un estimador del MSE, esto es, minimizar el Un-Biased Risk Estimator

$$\mathcal{V}_u(\lambda) = \|y - Ay\|^2/n - \sigma^2 + 2tr(A)\sigma^2/n, \tag{1.84}$$

que es también el C_p de Mallow. Si σ^2 es conocida, estimar λ minimizando $\mathcal{V}_u$ funciona bien, en cambio, si hay que estimarla surgen problemas. Por ejemplo, sustituyendo la aproximación

$$\mathbb{E}(\|y - Ay\|^2) = \hat{\sigma}^2(n - tr(A)), \tag{1.85}$$

siendo

$$\hat{\sigma}^2 = \frac{\|y - Ay\|^2}{n - tr(A)} \tag{1.86}$$

y, sustituyendo en la ecuación (1.83)

$$M = \mathbb{E}(\|\mu - X\hat{\beta}\|^2/n) = \frac{tr(A)}{n}\sigma^2 \tag{1.87}$$

y el estimador MSE $\tilde{M} = tr(A)\hat{\sigma}^2/n$. El problema de este método es que, como se demuestra en [39], no se puede utilizar para la selección de modelos porque influye en el resultado el número de parámetros.

b) Cross Validation

Alternativamente al método anterior, la estimación del parámetro de suavizado puede basarse en el error cuadrático medio de predicción:

$$P = \sigma^2 + M. \tag{1.88}$$

Como este criterio P depende directamente de σ^2, es mucho más resistente a la sobreestimación que el criterio basado en M solamente.

La forma más obvia de estimar P es utilizando validación cruzada. Omitiendo un dato y_i del proceso de ajuste del modelo, adquiere independencia del ajuste del modelo del resto de datos. Entonces, el error cuadrático de la predicción de y_i es fácilmente estimado y, extrayendo en cada caso un y_i distinto, llegamos al método de estimación cross validation ordinario de P:

$$\mathcal{V}_0 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\mu}_i^{[-i]})^2, \tag{1.89}$$

donde $\hat{\mu}_i^{[-i]}$ denota la predicción de $\mathbb{E}(y_i)$ obtenida con el ajuste del modelo de todos los datos excepto y_i. Pero este proceso es bastante problemático ya que deberíamos realizar n ajustes del modelo. Por eso, [39] desarrolla $\mathcal{V}_0$ para obtener un cálculo más sencillo, que es

$$\mathcal{V}_0 = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{\mu}_i)^2}{(1 - A_{ii})^2}. \tag{1.90}$$

Este se puede calcular realizando un solo ajuste del modelo original.

Aun así, este método tiene un gran problema: es computacionalmente intensivo minimizar en el caso de modelos aditivos.

c) Generalized Cross Validation

En este caso, para obtener el λ por validación cruzada generalizada [8], se minimiza la siguiente versión de la función $\mathcal{V}(\lambda)$ en forma matricial:

$$\mathcal{V}(\lambda) = \frac{1}{n}\|(I - A(\lambda))u\|^2 / \left[\frac{1}{n}tr(I - A(\lambda)).\right]^2 \tag{1.91}$$

1.4.2. Ajuste del modelo

El modelo aditivo tiene la forma

$$Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \varepsilon, \tag{1.92}$$

donde el error ε tiene media cero. Dadas observaciones (x_i,y_i), especificamos la suma de cuadrados penalizada como

$$PRSS(\alpha, f_1, f_2, ..., f_p) = \sum_{i=1}^{N} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(X_j)^2 dX_j, \tag{1.93}$$

donde $\lambda_j \geq 0$ es el parámetro de suavización. Este parámetro es el que controla el trade off entre el ajuste del modelo y la suavización. Cuando $\lambda \to \infty$, la estimación de f es una linea recta, mientras que, si $\lambda = 0$, resulta en una estimación sin penalizar.

Sin más restricciones en el modelo, el resultado de minimizar (1.93) no tiene una única solución, ya que podríamos sumar o restar una constante de cualquier función f_j y en función de eso ajustar α. Por eso, habitualmente se asume que $\sum_1^N f_j(x_{ij}) = 0 \quad \forall j$. Es fácil demostrar que en este caso $\hat{\alpha} = \mu(y_i)$. Si además de esta restricción la matriz de inputs es no singular, entonces (1.93) es estrictamente convexo y el problema de minimización es de respuesta única. En cambio, si la matriz es singular, entonces la parte lineal de las componentes f_j no pueden ser determinadas de forma única.

Podemos obtener la solución mediante un procedimiento iterativo simple. Fijando $\hat{\alpha} = \mu(y_i)$ aplicamos un spline cúbico suavizado S_j a la función objetivo $\{y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})\}_1^N$ para obtener un nuevo estimador $\hat{f}_j$. Se realiza este proceso para cada predictor utilizando la estimación actual de las otras funciones $\hat{f}_j$ y se continúa hasta que la estimación $\hat{f}_j$ se estabiliza. Este algoritmo se conoce como "backfitting", y el resultado es análogo a la regresión múltiple para modelos lineales. A continuación se especifica este algoritmo:

1. Inicializar: $\hat{\alpha} = \frac{1}{N} \sum_1^N y_i, \hat{f}_j \equiv 0, \forall i, j.$
2. Iteración: $j = 1, 2, ..., p, ..., 1, 2, ...p, ...$

$$\hat{f}_j \leftarrow S_j \left[\{y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})\}_1^N \right], \tag{1.94}$$

$$\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N} \sum_{i=1}^{N} \hat{f}_j(x_{ij}). \tag{1.95}$$

hasta que la función $\hat{f}_j$ cambie menos que un límite preestablecido.

A continuación estudiaremos estos modelos aditivos generalizados para el caso de la regresión logística.

1.4.3. Regresión logística aditiva

En el caso de un modelo de regresión logística para datos con respuesta binaria, relacionábamos la media de la respuesta $\pi(X) = P(Y = 1|X)$ con los predictores mediante un modelo de regresión lineal y la función link *logit*:

$$\log\left(\frac{\pi(X)}{1-\pi(X)}\right) = \alpha + \beta_1 X_1 + ... + \beta_p X_p. \tag{1.96}$$

El modelo de regresión logística aditivo es un ejemplo de modelo aditivo generalizado. Éste reemplaza cada término lineal por una forma funcional más general

$$\log\left(\frac{\pi(X)}{1-\pi(X)}\right) = \alpha + f_1(X_1) + ... + f_p(X_p). \tag{1.97}$$

Las funciones $f_1, f_2, ..., f_p$ se estiman mediante el algoritmo backfitting con el procedimiento de Newton-Raphson que se muestra a continuación:

1. Calcular los valores iniciales: $\hat{\alpha} = \log[\overline{y}/(1-\overline{y})]$, donde $\overline{y} = \mu(y_i)$, la proporción de unos de la muestra, y fijar $\hat{f}_j \equiv 0 \quad \forall j$.
2. Definir $\hat{\eta}_i = \hat{\alpha} + \sum_j \hat{f}_j(x_{ij})$ y $\hat{p}_i = 1/[1 + exp(-\hat{\eta}_i)]$. Iteración:
   a) Construir la variable objetivo

   $$z_i = \hat{\eta}_i + \frac{(y_i - \hat{p}_i)}{\hat{p}_i(1-\hat{p}_i)}$$

   .
   b) Construir los pesos $w_i = \hat{p}_i(1-\hat{p}_i)$.
   c) Ajustar un modelo aditivo para el objetivo z_i con pesos w_i utilizando un algoritmo backfitting ponderado. Esto da nuevas estimaciones $\hat{\alpha}, \hat{f}_j, \quad \forall j$.
3. Realizar el paso 2 hasta que el cambio en la función se encuentre por debajo de un límite preestablecido.

2. Modelos de regresión en alta dimensión

En esta sección, en primer lugar, hablaremos brevemente sobre los problemas causados por la situación en la que $p \gg N$. A continuación, describiremos tres tipos de metodologías para llevar a cabo el estudio estadístico de datos en alta dimensión, cuando la variable respuesta es continua, además de extender los modelos GAM a esta situación. Pero, en algunos de los casos, aplicaremos estos modelos al caso de variable respuesta binaria. Estos tres tipos de metodología son:

- **Selección de subconjuntos.** Consiste en identificar un subconjunto de covariables que se cree están relacionadas con la variable respuesta y ajustar un modelo con ellas.
- **Regularización.** Se ajusta un modelo con todas las covariables, pero se reducen los coeficientes estimados hacia cero. Esta regularización puede aplicar selección de subconjuntos.
- **Reducción de la dimensión.** Se proyectan las p covariables en un subespacio M-dimensional donde $M < p$. Esto se realiza mediante M convinaciones lineales diferentes, o proyecciones, de las variables. Entonces, esas M proyecciones son utilizadas como predictores para ajustar el modelo de regresión.

2.1. Problemas en alta dimensión

Cuando $p > N$, por ejemplo, el método de mínimos cuadrados, no se puede llevar a cabo. La razón es que, independientemente de que exista o no relación entre las covariables y la respuesta, mínimos cuadrados ajustará de forma perfecta los datos, por lo que los residuos serán cero.

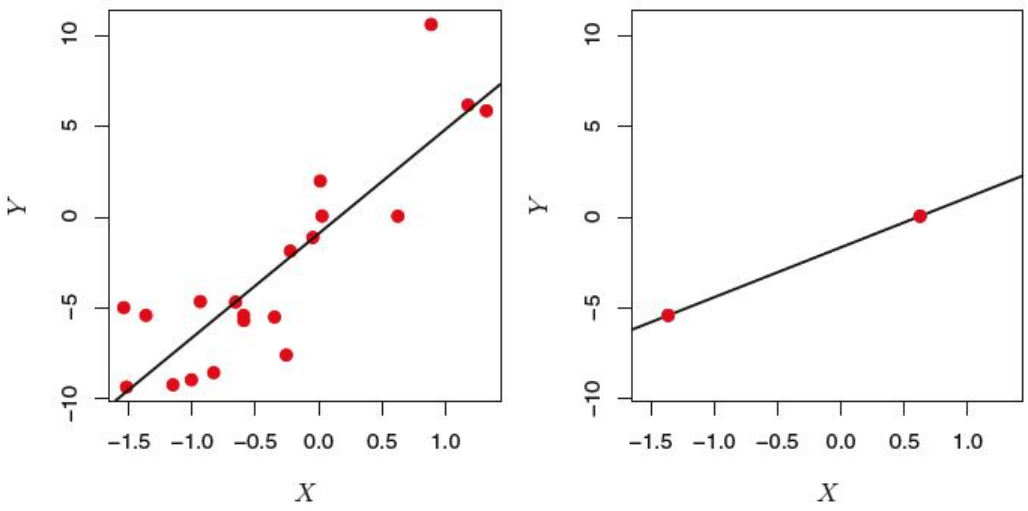

Gráfico 1: Izquierda: regresión por mínimos cuadrados con 20 observaciones. Derecha: regresión por mínimos cuadrados con 2 observaciones. Figura obtendia de [23].

En la Figura 1 se pueden ver dos gráficos en los que para la estimación de la recta de regresión se realizó mínimos cuadrados con una sola covariable más el intercepto. En el gráfico de la izquierda se utilizaron 20 observaciones, mientras que en el de la derecha se utilizaron 2. En el caso de 20 observaciones, $n > p$ y la linea de regresión no ajusta perfectamente los datos. En cambio, en el caso de 2 observaciones, ajusta perfectamente los datos. Esto es un problema ya que al ajustar perfectamente, seguramente nos lleve a sobreestimar los datos. Es decir, ajusta perfectamente los datos de entrenamiento, pero va a ser un mal modelo a la hora de predecir nuevos datos.

Existen tres puntos importantes a la hora de enfrentarse a un problema de regresión en alta dimensión: (1) la regularización o reducción de covariables juegan un papel fundamental; (2) la elección del parámetro de reducción correcto; y (3) el error de predicción tiende a crecer cuando aumenta la dimensión del problema, a menos que las covariables del modelo realmente tengan relación con la varible respuesta.

El tercer punto es quizás el más importante y es conocido como la maldición de la dimensión. Se podría pensar que al aumentar el número de covariables en el modelo, la calidad del ajuste va a mejorar pero, como se demuestra en [23], eso solo ocurre si las covariables realmente están relacionadas con la variable respuesta, en caso contrario, empeorará el ajuste del modelo aumentando el riesgo de sobreestimación y aumentará el error de predicción.

En los modelos de regresión de baja dimensión, debemos tener cuidado con la multicolinealidad pero, en problemas de alta dimensión, este problema se agraba. La multicolinealidad consiste en que al menos una covariable del modelo es combinación lineal de otras covariables. Esto provoca que no podamos saber que covariables realmente predicen la variable respuesta ni identificar los mejores coeficientes para la regresión.

2.2. Modelos de selección de subconjuntos

Estos métodos consisten en seleccionar las variables que vamos a utilizar en el modelo, mientras que eliminamos el resto. Existen varios métodos para seleccionar las covariables que mantenemos en el modelo, con la intención de reducir la varianza de la predicción y mejorar su interpretabilidad.

2.2.1. Selección del mejor subconjunto

Siendo $\mathcal{M}_0$ el modelo con solamente el intercepto y variable respuesta continua, para $k = 1, 2, ..., p$, ajustamos todos los $\binom{p}{k}$ modelos que contienen exactamente k predictores; a continuación, escogemos el mejor de esos modelos, es decir, el que tenga, por ejemplo, una menor RSS o un mayor R^2, y se nombra como $\mathcal{M}_k$. Por último, se selecciona un modelo de entre $\mathcal{M}_0, ..., \mathcal{M}_p$

utilizando cualquier criterio de selección de modelos.

Pero esta técnica tiene varias dificultades a la hora de aplicarlo. El principal es que cuando el número de covariables p es muy elevado, el espacio de búsqueda es muy grande, por lo que hay muchas posibilidades de encontrar modelos que se ajusten bien a la muestra de entrenamiento pero que no sean buenos prediciendo. Este gran espacio de búsqueda puede llevar a sobreestimar y a que los coeficientes estimados tengan varianzas altas, además de que el tiempo de computación va a ser muy elevado.

Cuando la variable respuesta es binaria, necestiamos hacer ciertas transformaciones para poder llevar a cabo el proceso de selección del mejor subconjunto [33].

Como ya comentamos, la log-odds de $Y = 1$ para un X dado se puede modelar como una función lineal de la forma:

$$E(Y = 1|X) = \ln\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_{k-1} x_{(k-1)i}. \tag{2.1}$$

Como las observaciones son una variable aleatoria de Bernoulli y, asumimos que son independientes, la función de log-verosimilitud se define como:

$$L(\beta) = \sum_{i=1}^{n} Y_i \ln\left(\frac{\pi_i}{1-\pi_i}\right) + \sum_{i=1}^{n} \ln(1-\pi_i). \tag{2.2}$$

Maximizando esta función, obtenemos los $\hat{\beta}$. En el caso de regresión logística múltiple, con Y denotando el vetor de respuestas y θ denotando $E(Y)$, la ecuación de verosimilitud se puede escribir (en notación matricial) como:

$$\frac{\partial L(\beta)}{\partial \beta} = X'(Y - \theta). \tag{2.3}$$

Igualamos esta ecuación a 0, $\frac{\partial L(\beta)}{\partial \beta} = 0$, por lo que $\hat{Y} = \hat{\theta}$, y satisface $X'(Y - \hat{Y}) = 0$. Realizando esta operación para todos los β obtenemos

$$\hat{\beta} = (X'WX)^{-1}X'WZ, \tag{2.4}$$

con W la matriz nxn diagonal con elementos $w_i = \hat{\theta}_i(1-\hat{\theta}_i)\hat{\theta}_i$, que es la probabilidad logística estimada, y $Z = X\hat{\beta} + W^{-1}\hat{\varepsilon}$ es el vector de observaciones de una variable pseudo-dependiente con $\hat{\varepsilon} = (Y - \hat{\theta})$ el vector de residuos. El elemento general de la variable pseudo-dependiente Z_i para el caso de pesos $w_i = \hat{\theta}_i(1-\hat{\theta}_i)$ es ([33])

$$Z_i = (1, x_i')\hat{\beta} + \frac{y_i - \hat{\theta}_i}{\hat{\theta}_i(1-\hat{\theta}_i)} = \hat{\beta}_0 + \sum_{j=1}^{k-1} \hat{\beta}_j x_{ij} + \frac{y_i - \hat{\theta}_i}{\hat{\theta}_i(1-\hat{\theta}_i)}. \tag{2.5}$$

La expresión de $\hat{\beta}$ dada en la ecuación (2.4), nos proporciona una base para utilizar un programa de regresión lineal. Utilizando los valores de Z_i como variable dependiente, los valores de

X_i como covariables y w_i como pesos, los coeficientes estimados obtenidos con un programa de regresión lineal van a ser los mismos que los estimadores por máxima verosimilitud obtenidos mediante un programa de regresión logística.

La suma de cuadrados de los residuos (RSS(k)) obtenida de el programa de regresión lineal es

$$RSS(k) = \sum_{i=1}^{n} w_i(z_i - \hat{z}_i)^2 = \sum_{i=1}^{n} \frac{(y_i - \hat{\theta}_i)^2}{\hat{\theta}_i(1-\hat{\theta}_i)}. \tag{2.6}$$

Esta RSS es igual que el estadístico χ^2 de Pearson de un programa de regresión logística. La suma de residuos al cuadrado media es $\hat{\sigma}^2 = \chi^2/n - k$. La estimación del error estándar de los coeficientes estimados mediante regresión lineal, son $\hat{\sigma}$ veces la raíz cuadrada de los elementos diagonales de la matriz $(X'WX)^{-1}$. Entonces, para obtener los valores correctos del error estándar de los estimadores de máxima verosimilitud, necesitamos dividir las estimaciones del error estándar de la regresión lineal por $\hat{\sigma}$. Esto nos permite utilizar cualquier programa de selección del mejor subconjunto lineal para realizar selección del mejor subconjunto logístico.

2.2.2. Selección forward-stepwise

Forward-Stepwise consiste en un modelo inicial en el que el único predictor es el intercepto y se van añadiendo una a una las covariables que mejoran en mayor medida el modelo.

Siendo $\mathcal{M}_0$ el modelo con solamente el intercepto. Para $k = 0, 1, ..., p-1$, se consideran todos los $p-k$ modelos que aumentan el número de predictores en $\mathcal{M}_k$ con una sola covariable adicional; entonces, se escoge el mejor de esos $p-k$ modelos, es decir, el que tenga una menor RSS, y se nombra como $\mathcal{M}_{k+1}$. Por último, se selecciona uno de los $\mathcal{M}_0, ..., \mathcal{M}_p$ modelos mediante algún criterio de selección de modelos.

2.2.3. Selección backward-stepwise

En cambio, Backward-Stepwise comienza con el modelo completo y va eliminando las covariables que menos impacto tienen sobre el ajuste una a una.

Siendo $\mathcal{M}_p$ el modelo completo con todos los predictores. Para $k = p, p-1, ..., 1$, se consideran los k modelos que contienen todos los predictores exceptuando uno en $\mathcal{M}_k$, es decir, contiene un total de $k-1$ predictores. Se escoge de esos k modelos el que tenga una menor RSS y lo llamamos $\mathcal{M}_{k-1}$. Por último, seleccionamos un modelo de entre $\mathcal{M}_0, ..., \mathcal{M}_p$ mediante un criterio de selección de modelos.

Tanto forward como backward-stepwise solamente estima $1 + p(p+1)/2$ modelos, por lo que se pueden aplicar cuando el número de covariables es muy alto. Pero estos dos métodos no aseguran

que encuentren el mejor modelo.

El criterio backward solamente se puede utilizar cuando $N > p$ mientras que Forward-Stepwise se puede utilizar siempre. Por eso, de los tres métodos aquí expuestos, la mejor opción para modelo con un número muy grande de covariables es el forward-stepwise.

Tanto en el caso de forward- como en el de backward-stepwise para el caso de modelos logísticos, el procedimiento es el mismo que para el modelo lineal [33]. La única diferencia es el criterio que se utiliza para incluir o excluir variables. En el caso del modelo logístico, uno de los criterios más utilizados es el test χ^2. Utilizando este test, incluiremos (excluiremos) la variable que produce un cambio mayor en la log-verosimilitud con respecto al modelo sin (con) esa variable. El mayor problema de este método es decidir cual será el nivel escogido para eliminar o incluir las variables. Algunos autores ([31] por ejemplo) sugieren que el nivel de 0.05 es demasiado restrictivo, por eso se proponen como niveles 0.15 para la entrada de una variable, y 0.20 para la salida.

2.2.4. Regresión forward-stagewise

Este método comienza del mismo modo que el método forward-stepwise, con solamente el intercepto y los coeficientes del resto de predictores recentrados igualados a cero. A cada paso, el algoritmo identifica la variable más correlada con los residuos del modelo. Entonces, estima los residuos de la regresión lineal simple en esa variable y la añade al modelo. Este proceso continua hasta que ninguna de las covariables tiene correlación con los residuos.

A diferencia del método forward-stepwise, ninguna otra variable se estima cuando un término se añade al modelo. Como consecuencia, forward-stagewise puede durar más de p pasos para alcanzar el ajuste por mínimos cuadrados. A pesar de ser un método más lento, es muy competitivo en problemas de alta dimensión.

2.2.5. Criterios de selección de modelos

En los métodos de backward- y forward-stepwise se añadía o eliminaba una variable del modelo de acuerdo a la significación de los coeficientes. Pero, en lugar de utilizar la significación, se puede construir una medida global de cada modelo que tenga en cuenta el ajuste a la vez que compense el exceso de parámetros. El objetivo es escoger aquel modelo que tenga una mejor medida global.

Cuatro de los criterios más utilizados son el R^2 ajustado, el Criterio de Información de Akaike (AIC)[10], el Criterio de Información de Bayes (BIC)[34] y el C_p de Mallows [27].

El coeficiente de determinación ajustado (R^2 ajustado) se define como

$$R^2\text{ajustado} = 1 - \frac{RSS/(n-p-1)}{TSS/(n-1)}, \tag{2.7}$$

siendo p los grados de libertad de RSS, n el tamaño muestral y TSS la suma de cuadrados totales, que se define como

$$TSS = \sum_{i=1}^{n} (Y_i - \overline{Y})^2. \tag{2.8}$$

El R^2 ajustado toma valores en el intervalo $[0, 1]$, significando el 1 que el modelo explica totalmente la varianza y 0 que no explica nada. Al dividir en la ecuación (2.7) por los grados de libertad se obtiene una comparación más justa, ya que tiene en cuenta la complejidad del modelo.

Los otros dos criterios, AIC y BIC, se definen como

$$AIC = -2\log(L) + 2p \tag{2.9}$$

$$BIC = -2\log(L) + p\log(n), \tag{2.10}$$

siendo L el máximo de la función de verosimilitud del modelo y p el número de parámetros. La definición de estos dos métodos se basa en la verosimilitud ya que así se pueden utilizar en todo tipo de modelos y no solamente en los modelos lineales.

El objetivo es encontrar un modelo cuyo AIC o BIC sea pequeño, ya que implicaría una verosimilitud grande y pocos parámetros. Pero estos dos términos están contrapuestos, por eso, el objetivo es encontrar modelos que incorporen solamente las variables útiles para reducir el número de parámetros a la vez que aumenta la verosimilitud. Con útiles nos referimos a que produzcan una gran reducción en la RSS.

Otro criterio de selección de modelos es el C_p de Mallows, introducido por Mallows en 1973 [27]. Su expresión es

$$C_p = \frac{RSS_p}{\hat{\sigma}_c^2} - n + 2p, \tag{2.11}$$

siendo RSS_p la suma de los residuos al cuadrado del modelo con p regresores y $\hat{\sigma}_c^2$ el estimador de la varianza del término de error del modelo completo. El criterio es que se escoge el modelo con un C_p pequeño. Este es un criterio muy parecido al AIC.

2.3. Métodos de regularización

Estos métodos ajustan los modelos con las p covariables pero aplicando restricciones o regularizaciones a los estimadores de los coeficientes. Así, conseguimos reducir significativamente la varianza de los estimadores. Los métodos más utilizados son regresión Ridge, Lasso y least angle regression. Extenderemos estos modelos al problema de clasificación con los modelos logísticos.

2.3.1. Regresión ridge

Este modelo de regresión reduce los coeficientes de la regresión mediante penalizaciones en su tamaño. Los coeficientes minimizan la RSS penalizada:

$$\hat{\beta}^{ridge} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{2.12}$$

$\lambda \geq 0$ controla el nivel de penalización. Cuanto mayor sea λ mayor será la penalización. Si $\lambda = 0$, obtendremos los estimadores mínimo cuadrados, en cambio, si $\lambda \to \infty$, los estimadores tenderán a cero. Por eso, según el λ que escojamos los estimadores $\hat{\beta}^{ridge}$ serán distintos.

Vemos que λ no afecta a β_0 ya que si lo penalizamos provocaríamos que el proceso dependiera del origen escogido para Y. Los coeficientes de este modelo no son invariantes ante cambios de escala en las covariables, por lo que normalmente se estandarizan las variables.

Una forma equivalente de presentar el problema de regresión ridge es:

$$\begin{aligned} \hat{\beta}^{ridge} = \arg\min_{\beta} & \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2, \\ \text{sujeto a } & \sum_{j=1}^{p} \beta_j^2 \leq t. \end{aligned} \tag{2.13}$$

Existe una correspondencia directa entre t y λ de las ecuaciones (2.13) y (2.12), respectivamente.

Para entender mejor el funcionamiento de la regresión ridge, utilizaremos la descomposición en valores singulares (SVD, Singular Value Decomposition) de la matriz de inputs centrados X. Esta descomposición es de la forma

$$X = UDV', \tag{2.14}$$

donde U y V son matrices ortogonales Nxp y pxp, respectivamente, con las columnas de U formando una base ortonormal del subespacio generado por las columnas de X, y las columnas de V una base ortonormal del subespacio generado por las filas de X. D es una matriz diagonal pxp, cuyos valores son $d_1 \geq d_2 \geq ... \geq d_p \geq 0$ llamados valores singulares de X. Si uno o más valores $d_j = 0$, X es singular. Utilizando esta descomposición, el vector ajustado por mínimos cuadrados quedaría:

$$\begin{aligned} X\hat{\beta} &= X(X'X)^{-1}X'y \\ &= UU'y. \end{aligned} \tag{2.15}$$

$U'y$ son las coordenadas de y con respecto a la base ortonormal U. La solución ridge aplicando

esta descomposición será:

$$\begin{aligned} X\hat{\beta}^{ridge} &= X(X'X+\lambda I)^{-1}X'y \\ &= UD(D^2+\lambda I)^{-1}DU'y \\ &= \sum_{j=1}^{p} u_j \frac{d_j^2}{d_j^2+\lambda} u_j' y, \end{aligned} \tag{2.16}$$

donde u_j son las columnas de U. Si $\lambda \geq 0$, $d_j^2/(d_j^2+\lambda) \leq 1$. Igual que sucede en la regresión lineal, la regresión ridge computa las coordinadas de y con respecto a la base ortonormal U, y estas coordenadas son reducidas por el factor $d_j^2/(d_j^2+\lambda)$. Esto implica que cuanto menor sea d_j^2, mayor será la reducción.

Para entender lo que significa que d_j^2 sea pequeño debemos relacionar la descomoposición en valores singulares con las componentes principales (apartado 2.4.1). La descomposición de la matriz centrada X es otra forma de expresar las componentes principales de las variables en X [16]. Con la matriz de covarianzas $S = X'X/N$ y la ecuación (2.14), tenemos

$$X'X = VD^2V', \tag{2.17}$$

la descomposición espectral de $X'X$. Los autovectores v_j (columnas de V) también son conocidos como componentes principales. La primera componente principal tiene dirección v_1 que tiene la propiedad de que $z_1 = Xv_1$ es la combinación lineal normalizada de las columnas de X con mayor varianza. Esta varianza es de la forma

$$Var(z_1) = Var(Xv_1) = \frac{d_1^2}{N} \tag{2.18}$$

y, de hecho, $z_1 = Xv_1 = u_1 d_1$. Como z_1 es la primera componente principal de X, u_1 es la primera componente principal normalizada. Entonces, la componente principal z_j tiene como máximo una varianza de d_j^2/N. Como consecuencia, la última componente principal tiene la menor varianza. Por lo tanto, los valores singulares d_j más pequeños se corresponden con las direcciones de menor varianza del espacio de columnas de X, y la regresión ridge reducirá en mayor medida esas direcciones.

Para estimar el modelo, en primer lugar estimamos β_0 como $\overline{y} = \frac{1}{N}\sum_{y=1}^{N} y_i$. El resto de coeficientes se estima mediante ridge regression pero sin intercepto utilizando x_{ij} centrados $(x_{ij} - \overline{x}_j)$. Entonces, de forma matricial, escribimos la ecuación (2.12) como

$$RSS(\lambda) = (y - X\beta)'(y - X\beta) + \lambda\beta'\beta \tag{2.19}$$

y la solución es

$$\begin{aligned} \hat{\beta}^{ridge} &= \left(X'X + \lambda I_p\right)^{-1} X'y \\ &= WX'y, \end{aligned} \tag{2.20}$$

donde I_p es la matriz identidad p x p. Se puede demostrar que, en caso de ser las covariables ortogonales, las $\hat{\beta}^{ridge}$ son versiones escaladas de los estimadores por mínimo cuadrados: $\hat{\beta}^{ridge} = \hat{\beta}/(1+\lambda)$.

El resultado (2.20) se puede expresar como

$$\begin{aligned} \hat{\beta}^{ridge} &= (I_p + \lambda(X'X)^{-1})^{-1}\hat{\beta} \\ &= Z\hat{\beta} \end{aligned} \tag{2.21}$$

Para poder estudiar $\hat{\beta}^{ridge}$ desde el punto de vista del error cuadrático medio, es necesario obtener la expresión de $\mathbb{E}[L_1^2(\lambda)]$ [21]:

$$\begin{aligned} \mathbb{E}[L_1^2(\lambda)] &= \mathbb{E}\left[(\hat{\beta}^{ridge} - \beta)'(\hat{\beta}^{ridge} - \beta)\right] \\ &= \mathbb{E}\left[(\hat{\beta} - \beta)'Z'Z(\hat{\beta} - \beta)\right] + (Z\beta - \beta)'(Z\beta - \beta) \\ &= \sigma^2 \text{traza}(X'X)^{-1}Z'Z + \beta'(Z - I)'(Z - I)\beta \\ &= \sigma^2\left[\text{traza}(X'X + \lambda I)^{-1} - \lambda \text{traza}(X'X + \lambda I)^{-2}\right] + \lambda^2\beta'(X'X + \lambda I)^{-2}\beta \\ &= \sigma^2 \sum_1^p \Lambda_i/(\Lambda_i + \lambda)^2 + \lambda^2\beta'(X'X + \lambda I)^{-2}\beta \\ &= \gamma_1(\lambda) + \gamma_2(\lambda) \end{aligned} \tag{2.22}$$

siendo Λ los autovalores de la matriz $X'X$. El elemento $\gamma_2(\lambda)$ es la distancia cuadrada de $Z\beta$ a β. Esta distancia será cero cuando $\lambda = 0$, ya que Z sería igual a I. $\gamma_1(\lambda)$ se puede considerar el cuadrado de la bias que se introduce cuando se utiliza $\hat{\beta}^{ridge}$ en lugar de $\hat{\beta}$. Puede verse como la suma de la varianza de los parámetros estimados. En términos de la variable aleatoria Y,

$$\hat{\beta}^{ridge} = Z\hat{\beta} = Z(X'X)^{-1}X'y. \tag{2.23}$$

Entonces,

$$\begin{aligned} VAR(\hat{\beta}^{ridge}) &= Z(X'X)^{-1}X'Var(y)X(X'X)^{-1}Z' \\ &= \sigma^2 Z(X'X)^{-1}Z'. \end{aligned} \tag{2.24}$$

La suma de las varianzas de todos los $\hat{\beta}^{ridge}$ es la suma de los elementos diagonales de (2.24).

La relación existente entre la varianza, la bias al cuadrado y el parámetro λ es que, cuando aumenta λ, la varianza total decrece mientras que la bias al cuadrado se incrementa. En [21] se demuestra que existe la posibilidad de que para ciertos valores de λ, el error cuadrático medio de $\hat{\beta}^{ridge}$ sea menor que el de $\hat{\beta}$. El valor de las derivadas de las funciones de la ecuación (2.22) al rededor del origen son

$$\lim_{\lambda \to 0^+} (\partial\gamma_1/\partial\lambda) = -2\sigma^2 \sum (1/\Lambda_i^2) \tag{2.25}$$

$$\lim_{\lambda \to 0^+} (\partial\gamma_2/\partial\lambda) = 0. \tag{2.26}$$

Entonces, $\gamma_1(\lambda)$ tiene una derivada negativa, la cual tiende a $-2p\sigma^2$ cuando $\lambda \to 0^+$ para $X'X$ ortogonal y tiende a $-\infty$ cuando $X'X$ es singular y $\Lambda_p \to 0$. Por otro lado, cuando $\lambda \to 0^+$, $\gamma_2(\lambda)$ es cero en el origen. Estas propiedades nos llevan a concluir que es posible fijar $k > 0$, lo que permite aumentar un poco la bias a la vez que se reduce substancialmente la varianza, mejorando así el error cuadrático medio de la estimación y la predicción.

Los grados de libertad efectivos del ajuste de regresión ridge son ([16]):

$$\begin{aligned} df(\lambda) &= tr[X(X'X + \lambda I)^{-1}X'] \\ &= tr(H_\lambda) \\ &= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda} \end{aligned} \tag{2.27}$$

que, como se puede ver, es una función monótona decreciente de λ. Normalmente, en los ajustes de regresión lineales, los grados de libertad son p, el número de parámetros, pero ya que algunos de los parámetros de la regresión ridge son cero, los grados de libertad serán menores. Cuando $\lambda = 0$, $df(\lambda) = p$, mientras que cuando $\lambda \to \infty$, $df(\lambda) \to 0$. Hay que recordar que el intercepto se ha eliminado a priori, por lo que a estos grados de libertad habrá que sumarle uno.

En el caso de regresión con variable respuesta binaria, para estimar los parámetros β, debemos maximizar, como ya comentamos anteriormente, la función de log-verosimilitud, que es de la forma

$$l(\beta) = \sum_i [Y_i \log p(X_i) + (1 - Y_i) \log(1 - p(X_i))]. \tag{2.28}$$

Pero si a la vez que estimamos los parámetros, queremos llevar a cabo regularización, debemos aplicarle una penalización. En este caso, utilizaremos la penalización en L_2 ([25]). Entonces, la función a maximizar es de la forma

$$l^\lambda(\beta) = l(\beta) - \lambda \|\beta\|^2, \tag{2.29}$$

donde $l(\beta)$ es la función de verosimilitud sin penalizar, y $\|\beta\|$ es la norma del vector de parámetros β. El resultado de maximizar esta función, lo denotaremos por $\hat{\beta}^\lambda$. Como en el caso continuo, el parámetro λ controla el nivel de la restricción. Cuando $\lambda = 0$, la solución será la obtenida para la regresión logística sin penalizar, mientras que si $\lambda \to \infty$, los β_j tenderán a cero.

La solución $\hat{\beta}^\lambda$ puede ser obtenida mediante el proceso de maximización de Newton-Raphson. La primera derivada de $l^\lambda(\beta)$ es

$$U^\lambda(\beta) = \sum_i X_i' \{Y_i - p(X_i)\} - 2\lambda\beta \tag{2.30}$$

$$= U(\beta) - 2\lambda\beta, \tag{2.31}$$

con $U(\beta)$ la derivada de la función de log-verosimilitud sin restricciones. La segunda derivada de la matriz negativa es

$$\Omega^\lambda(\beta) = \Omega(\beta) + 2\lambda I, \tag{2.32}$$

donde $\Omega = X'V(\beta)X$ es la matriz negativa de la segunda derivada de la verosimilitud sin restricciones y $V(\beta)$ es una matriz diagonal nxn con valores $v_{ii} = p(X_i)(1 - p(X_i))$.

Tras una serie de transformaciones (ver [25] para más detalles), podemos presentar la estimación de $\hat{\beta}^\lambda$ como

$$\hat{\beta}^\lambda = (\Omega(\beta_0) + 2\lambda I)^{-1}\Omega(\beta_0)\hat{\beta}. \tag{2.33}$$

Vemos que $\hat{\beta}^{\lambda}$ se reduce hacia cero si el valor del parámetro ridge (λ) se incrementa.

Para escoger el parámetro de suavizado λ, nos basaremos en el error de predicción. Este error se calcula estimando las probabilidades $\hat{p}(x)$. Predecimos para nuevas observaciones conocidas, X_{new}, la probabilidad de que $Y_{new} = 1$ con $\hat{p} = \hat{p}(X_{new})$ y denotando por p la probabilidad real de que $Y_{new} = 1$. Hay tres formas de medir este error, que son [25]:

- Error de clasificación:

$$\begin{aligned} CE &= 1 \qquad \text{si } Y_{new} = 1 \text{ y } \hat{p} < \frac{1}{2} \text{ o } Y_{new} = 0 \text{ y } \hat{p} > \frac{1}{2}, \\ &= \frac{1}{2} \qquad \text{si } \hat{p} = \frac{1}{2}, \\ &= 0 \qquad \text{en otro caso.} \end{aligned} \tag{2.34}$$

- Error cuadrático:

$$SE = (Y_{new} - \hat{p})^2. \tag{2.35}$$

- Error de log-verosimilitud mínimo:

$$ML = -\{Y_{new} \log \hat{p} + (1 - Y_{new}) \log(1 - \hat{p})\}. \tag{2.36}$$

La media de las tres medidas es máxima si p está alrededor de $\frac{1}{2}$ y tiende a cero si p tiende a 1 o a 0. La elección del método depende de como vayamos a utilizar el modelo para predecir. Si nuestra regla de predicción va a ser que $\hat{Y}_{new} = 1$ si $\hat{p} > \frac{1}{2}$, $\hat{Y}_{new} = 0$ si $\hat{p} < \frac{1}{2}$ y asignar 1 o 0 aleatoriamente en caso de que $\hat{p} = \frac{1}{2}$, la mejor medición del error será la primera (2.34).

El segundo método (2.35), mide la distancia euclídea entre Y_{new} y $\hat{p}$. Es una analogía directa del error cuadrático en los modelos de regresión lineales ordinarios.

La tercera medida (2.36), igual a $-\log \hat{p}$ si $Y_{new} = 1$ e igual a $-\log(1 - \hat{\beta})$ si $Y_{new} = 0$, es la típica medida utilizada en el caso de datos binarios. Sumando esta medida del error para todas las observaciones, nos lleva a menos la log-verosimilitud de los datos (X, Y), dado el vector de parámetros β. Las ventajas de este método son que están relacionadas con la función de log-verosimilitud y que no es un método exclusivo de la regresión logística, sino que se puede utilizar en otros modelos.

Una vez que hemos decidido la medida del error que vamos a utilizar, calcularemos ese error para distintos valores de λ y escogeremos aquel λ que lo minimice.

2.3.2. Lasso

Este es otro método de regularización muy parecido a la ridge regression pero con importantes diferencias. Este método realiza selección de variables y estimación simultaneamente. Lasso se

encuentra a medio camino entre los métodos de regresión ridge y best subset selection. Los estimadores lasso deben resolver

$$\begin{aligned} \hat{\beta}^{lasso} = \arg\min_{\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \\ \text{sujeto a } \sum_{j=1}^{p} |\beta_j| \leq t, \end{aligned} \tag{2.37}$$

con t el parámetro de regularización que se le aplica a los estimadores. Podemos escribir este mismo problema en forma Lagrangiana como

$$\hat{\beta}^{lasso} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}, \tag{2.38}$$

con λ como parámetro de regularización lasso.

Vemos que la penalización en L_2 $\sum_{j=1}^{p} \beta_j^2$ del problema de la ridge regression (ecuación 2.12) se sustituye en este caso por la penalización en L_1 $\sum_{j=1}^{p} |\beta_j|$. Debido a esto, en el caso de lasso, no va a existir una expresión cerrada para los β como sucedía en la ridge regression (ecuación 2.20), pero se puede obtener resolviendo un problema de programación cuadrática.

Fijando t suficientemente pequeño en la ecuación (2.37), se consigue que algunos de los coeficientes sean cero. Si t se escoge mayor que $t_0 = \sum_{j=1}^{p} \left| \hat{\beta}_j \right|$, los estimadores lasso serán los $\hat{\beta}_j$, siendo $\hat{\beta}_j$ los estimadores por mínimos cuadrados. Si $t < t_0$ todos los coeficientes se reducen hacia cero e incluso algunos podrían llegar a ser iguales a cero. En cambio, si por ejemplo fijamos $t = t_0/2$, la reducción de los coeficientes mínimo cuadrados es de aproximadamente el 50 %. Se debe escoger aquel parámetro t que mínimiza la estimación del error de predicción esperado.

La relación que existe entre los parámetros estimados mediante el modelo lasso y los estimados mediante regresión mínimo cuadrada es: $\hat{\beta}^{lasso} = sign(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$.

Para escoger t, en [36] se describen dos métodos: cross-validation y cross-validation generalizado. Ambos métodos se utilizan cuando no conocemos la distribución de las observaciones (X, Y), pero en la práctica con datos reales no se suele saber si conocemos o no la distribución por lo que se pueden utilizar los dos métodos en ambos casos.

Para el primer método, supongamos que

$$Y = m(X) + \varepsilon, \tag{2.39}$$

donde $\mathbb{E}(\varepsilon) = 0$ y $Var(\varepsilon) = \sigma^2$. El error mínimo cuadrado de la estimación $\hat{m}(X)$ está definido por

$$ME = \mathbb{E}(\hat{m}(X) - m(X))^2. \tag{2.40}$$

Una medida similar es el error de predicción de $\hat{m}(X)$ dado por

$$PE = \mathbb{E}(Y - \hat{m}(X))^2 = ME + \sigma^2. \tag{2.41}$$

Lasso es indexado en términos del parámetro normalizado $s = t/\sum \hat{\beta}_j^0$, y el error de predicción es estimado en un grid de valores de s en el intervalo $[0, 1]$. Se selecciona el valor $\hat{s}$ que minimiza el PE.

El segundo método puede ser deducido de la aproximación lineal a el estimador lasso. Escribimos la restricción $\sum |\beta| \leq t$ como $\sum \beta_j^2/|\beta| \leq t$. Entonces, escribimos la solución $\tilde{\beta}$ al problema con esta nueva restricción como el estimador de la ridge regression

$$\tilde{\beta} = (X'X + \lambda W^-)^{-1} X'y, \tag{2.42}$$

donde $W = diag(|\tilde{\beta}_j|)$ y W^- es su inversa generalizada. Entonces, el número de parámetros estimados en $\tilde{\beta}$ puede ser aproximado por

$$p(t) = tr(X(X'X + \lambda W^-)^{-1} X'). \tag{2.43}$$

Siendo $RSS(t)$ la suma de cuadrados residual de $\tilde{\beta}$ con restricción t, el estadístico cross-validation generalizada es

$$GCV(t) = \frac{1}{N} \frac{RSS(t)}{(1 - p(t)/N)^2}. \tag{2.44}$$

Para la elección de t en el caso de regresión ridge, se haría de forma análoga con los mismos dos métodos que en este caso.

A continuación presentaremos el método lasso para el caso en el que la variable respuesta sea binaria.

Como ya se comentó anteriormente, en el modelo de regresión logística, las probabilidades de cada clase se estiman a través de una función lineal de los predictores que, para el caso de solamente dos clases, es de la forma:

$$\begin{aligned} Pr(G = 1|x) &= \frac{1}{1 + e^{-(\beta_0 + x^T\beta)}}, \\ Pr(G = 2|x) &= \frac{1}{1 + e^{(\beta_0 + x^T\beta)}} \\ &= 1 - Pr(G = 1|x), \end{aligned} \tag{2.45}$$

lo que implica que

$$log \frac{Pr(G = 1|x)}{Pr(G = 2|x)} = \beta_0 + x^T\beta. \tag{2.46}$$

Entonces, se ajusta el modelo mediante máxima verosimilitud regularizada. Siendo $p(x_i) = Pr(G = 1|x_i)$ la probabilidad (2.45) para una observación i y unos valores particulares para los

parámetros (β_0, β), se maximiza la log verosimilitud penalizada ([17])

$$\max_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[\frac{1}{N} \sum_{i=1}^{N} \{ I(g=1) \log p(x_i) + I(g_i = 2) \log(1 - p(x_i)) \} - \lambda_\alpha(\beta) \right]. \tag{2.47}$$

Denotando $y_i = I(g_i = 1)$, la parte de la log verosimilitud de (2.47) se puede escribir como

$$l(\beta_0, \beta) = \frac{1}{N} \sum_{i=1}^{N} y_i(\beta_0 + x_i^T \beta) - \log(1 + e^{(\beta_0 + x_i^T \beta)}). \tag{2.48}$$

El algoritmo de Newton para maximizar la log verosimilitud (no penalizada) (ecuación 2.48) equivale al algoritmo de mínimos cuadrados reponderados iterativamente. Entonces, si los parámetros estimados son $(\tilde{\beta}_0, \tilde{\beta})$, realizamos una aproximación cuadrática a la log verosimilitud, la cual es

$$l_Q(\beta_0, \beta) = -\frac{1}{2N} \sum_{i=1}^{N} w_i(z_i - \beta_0 - x_i^T \beta)^2 + C(\tilde{\beta}_0, \tilde{\beta})^2, \tag{2.49}$$

donde

$$z_i = \tilde{\beta}_0 + x_i^T \tilde{\beta} + \frac{y_i - \tilde{p}(x_i)}{\tilde{p}(x_i)(1 - \tilde{p}(x_i))}, \tag{2.50}$$

$$w_i = \tilde{p}(x_i)(1 - \tilde{p}(x_i)), \tag{2.51}$$

y $\tilde{p}(x_i)$ es evaluado con los parámetros de cada paso. La actualización de Newton se obtiene minimizando l_Q. La aproximación que se propone en [17] es similar. Para cada λ, crean un bucle externo que computa la aproximación cuadrática l_Q para los parámetros $(\tilde{\beta}_0, \tilde{\beta})$. Entonces, utilizan coordinadas descendientes para resolver el problema mínimo cuadrado ponderado penalizado

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \{ -l_Q(\beta_0, \beta) + \lambda P_\lambda(\beta) \}. \tag{2.52}$$

Esto equivale a una secuencia de bucles anidados:

bucle exterior: decrecimiento de λ.
bucle intermedio: actualiza la aproximación cuadrática l_Q utilizando los parámetro $(\tilde{\beta}_0, \tilde{\beta})$.
bucle interior: aplica el algoritmo de coordinadas decrecientes en el problema de mínimos cuadrados ponderados penalizados (2.52).

2.3.3. Discusión: subset selection, regresión ridge y lasso

En caso de que la matriz de inputs X sea ortogonal, los tres métodos tienen soluciones explícitas. Cada modelo aplica una simple transformación a los estimadores mínimo cuadrados.

Como se muestra en la Figura 2, el método de regresión ridge realiza una reducción proporcional de los coeficienes de la regresión por mínimos cuadrados, mientras que el método lasso traslada

cada coeficiente por el factor λ. Ambos métodos se conocen como "soft-thresholding". En cambio, el método de selección del mejor subconjunto, iguala a cero todos los coeficientes más pequeños que el M-ésimo más grande. Este último método se conoce como "hard-thresholding".

Estimator	Formula
Best subset (size M)	$\hat{\beta}_j \cdot I[\text{rank}(\lvert\hat{\beta}_j\rvert \leq M)$
Ridge	$\hat{\beta}_j/(1+\lambda)$
Lasso	$\text{sign}(\hat{\beta}_j)(\lvert\hat{\beta}_j\rvert - \lambda)_+$

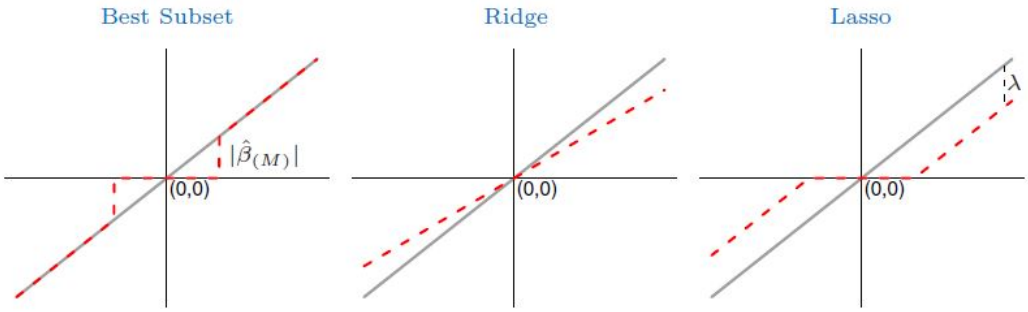

Gráfico 2: Estimadores de β_j en el caso de columnas de X ortonormales. M y λ son constantes escogidas por las técnicas correspondientes. En los gráficos se muestran los estimadores con líneas rojas y las líneas grises muestran las estimaciones sin restricciones.
Figura obtendia de [16].

En el caso no ortogonal, la relación entre los métodos lasso y regresión ridge se muestran en el Gráfico 3, que los representa con solamente dos parámetros. La suma de residuos al cuadrado tiene formas elípticas centradas en la estimación mínimo cuadrada. La región de restricción para lasso es $|\beta_1| + |\beta_2| \leq t$, mientras que para la regresión ridge es $\beta_1^2 + \beta_2^2 \leq t$. Ambos métodos encuentran el primer punto donde los contornos elípticos tocan la región de restricción. En el caso de lasso, cuya región tiene forma de diamante, si la elipse toca en una esquina, un parámetro β_j será igual a cero.

Podemos generalizar el criterio de optimización de ridge regression y lasso y verlo como estimadores bayesianos:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q \right\} \tag{2.53}$$

para $q \geq 0$. $q = 0$ se corresponde con la selección del mejor subconjunto, $q = 1$ con lasso y $q = 2$ con ridge regression. En el caso de $1 < q < 2$, se dan algunas extensiones propuestas por otros autores. $q = 1$ (lasso) es el q más pequeño cuya región de restricción es convexa. Las regiones

no convexas dificultan la resolución del problema de optimización.

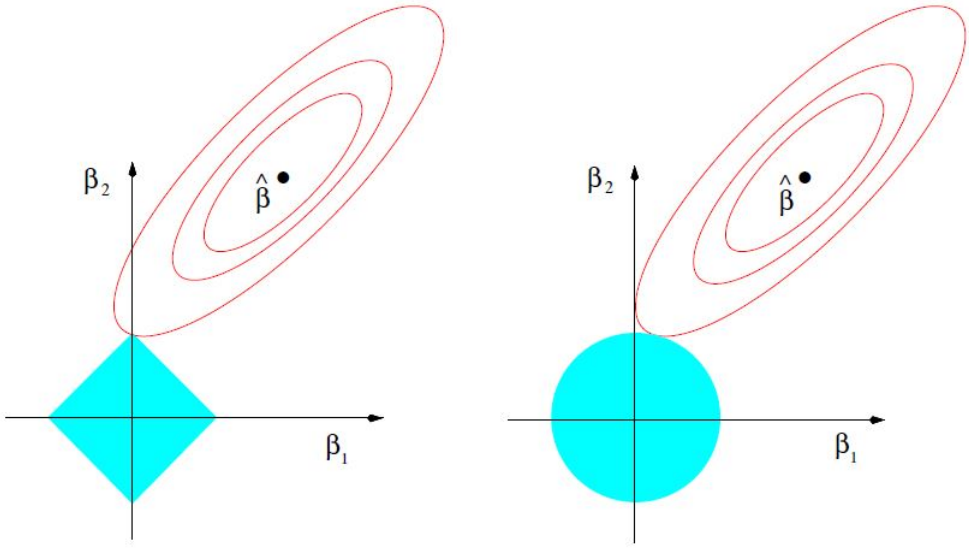

Gráfico 3: Estimación mediante lasso (izquierda) y regresión ridge (derecha). Las áreas azules se corresponden con las restricciones $|\beta_1| + |\beta_2| \leq t$ y $\beta_1^2 + \beta_2^2 \leq t^2$, respectivamente, mientras que las elipses moradas son los contornos de la función de error mínimo cuadrado.
Figura obtendia de [16].

2.3.4. Regresión least angle

Least Angle Regression (LAR) es una versión de la regresión forward-stagewise. La primera variable que se selecciona es aquella más correlada con la variable respuesta, entonces va variando el coeficiente de ésta hasta encontrar otra covariable que alcance el nivel de correlación con los residuos actuales que el que tiene la primera covariable con los mismos. La segunda variable se añade al conjunto de variables incluidas en el modelo y se varían los coeficientes de ambas conjuntamente de forma que se mantenga su correlación. Entonces, se añade la covariable que alcance el nivel de correlación con los residuos que tienen las otras covaribles. Este proceso continua hasta que todas las variables se encuentran en el modelo y termina con el ajuste completo por mínimos cuadrados. Por eso, solamente necesita p pasos, siendo p el número de covariables.

El algoritmo para llevar a cabo esta regresión es el siguiente:

1. Estandarizar los predictores. Comenzar con los residuos $r = y - \overline{y}, \beta_1, \beta_2, ..., \beta_p = 0$.
2. Encontrar el predictor x_j más correlado con r.
3. Mover β_j desde cero hacia el coeficiente mínimo cuadrado $< x_j, r >$ hasta que otro x_k tenga tanta correlación con los residuos actuales como x_j.

4. Mover β_j y β_k en la dirección de sus coeficientes mínimo cuadrados conjuntos de los residuos actuales en (x_j, x_k) hasta que otro predictor x_l tenga tanta correlación con los residuos actuales.
5. Repetir el paso 4 hasta que todos los p predictores estén incluidos. Después de mín$(N-1, p)$ pasos se obtiene la solución completa de mínimos cuadrados.

En el paso 5, si $p > N-1$, el algoritmo alcanza la solución de residuos cero después de N-1 pasos. De el algoritmo anterior podemos obtener el algoritmo para resolver el problema de lasso mediante una pequeña modificación.

4a. Si un coeficiente que no vale cero alcanza el cero, se elimina la variable correspondiente y se recomputa la dirección mínimo cuadrada.

Ambos algoritmos son muy eficientes, especialmente cuand $p \gg N$.

Supongamos que $\mathcal{A}_k$ es el conjunto de variables activas al comienzo del k-ésimo paso, y $\beta_{\mathcal{A}_k}$ es el vector de coeficientes para esas variables en este paso. Entonces, habrá $k-1$ valores distintos de cero, y el que entra será cero. Si $r_k = y - X_{\mathcal{A}_k}\beta_{\mathcal{A}_k}$ son los resiudos actuales, entonces la dirección en este paso será

$$\delta_k = \left(X_{\mathcal{A}_k}^T X_{\mathcal{A}_k}\right)^{-1} X_{\mathcal{A}_k} r_k. \tag{2.54}$$

Los coeficientes entonces pasan a ser $\beta_{\mathcal{A}_k}(\alpha) = \beta_{\mathcal{A}_k} + \alpha_k$. La dirección en este paso cumple que mantiene las correlaciones atadas.

Este modelo se puede extender a modelos logísticos. Considerando la log-verosimilitud logística para una función de regresión $f(x)$ la cual será lineal en x:

$$l(f) = \sum_{i=1}^{N} y_i f(x_i) - \log(1 + exp(f(x_i))). \tag{2.55}$$

Podemos inicializar $f(x) = \log(\overline{y}/(1-\overline{y}))$. Para algún α, deberemos encontrar la covariable x_j que aumente en mayor medida la log-verosimilitu logística, $l(f(x) + x_j^T\alpha)$. Para encontrar esta x_j, podemos computar la derivada direccional para cada j y elegir el máximo

$$\begin{aligned} j* &= \arg\max_j \left| \frac{d}{d\alpha} l(f(x) + x_j^T\alpha) \right|_{\alpha=0} \\ &= \arg\max_j \left| x_j^t \left(y - \frac{1}{1 + exp(-f(x))} \right) \right|. \end{aligned} \tag{2.56}$$

La covariable seleccionada es el primer miembro del conjunto activo A. Para α suficientemente pequeño, la ecuación (2.56) implica que

$$(s_{j*}x_{j*} - s_j x_j)^T \left(y - \frac{1}{1 + exp(-f(x) - x_{j*}^T\alpha)} \right) \geq 0, \tag{2.57}$$

para todo $j \in A^C$, donde s_j indica el signo de la correlación. Si escogemos α de forma que tengamos la máxima magnitud mientras mantenemos la restricción (2.57), obtendríamos un problema

de optimización no lineal. Sin embargo, linealizar (2.57) nos lleva a una buena aproximación. Si x_2 es la variable con la segunda mayor correlación con los residuos, entonces

$$\hat{\alpha} = \frac{(s_{j*}x_{j*} - s_2x_2)^T(y - p(x))}{(s_{j*}x_{j*} - s_2x_2)^T(p(x)(1 - p(x))x_{j*})}. \tag{2.58}$$

El algoritmo puede necesitar iterar la ecuación (2.58) para obtener el $\hat{\alpha}$ exacto. Y la misma lógica sigue el algoritmo para la solución completa.

2.4. Modelos de reducción de la dimensión

En esta sección estudiaremos métodos para resolver la situación en la que tenemos un gran número de covariables, normalmente muy correladas. Estes métodos consisten en producir una pequeña cantidad de combinaciones lineales de los inputs originales.

2.4.1. Regresión de componentes principales

Este método consiste en encontrar combinaciones lineales de las variables originales que representen lo mejor posible la variabilidad de los datos. Estas combinaciones lineales son lo que se conocen como componentes principales. La primera componente principal de x se define como una variable aleatoria z_1 que cumple

$$z_1 = v_1'x = v_{11}x_1 + ... + v_{d1}x_d \quad \text{con} \quad v_1 = (v_{11}, ..., v_{d1})' \in \mathcal{R}^d,$$

$$Var(z_1) = \text{máx}\{var(v'x) : v \in \mathcal{R}^d, v'v = 1\}, \tag{2.59}$$

siendo $x = (x_1, ..., x_d)$ un vector aleatorio d-dimensional con vector de medias $\mu = E(x)$ y matriz de covarianzas $\Sigma = E((x - \mu)(x - \mu)')$. Esta primera componente principal es la combinación lineal normalizada de mayor varianza de las variables de X.

Entonces, la primera componente $z_1 = v_1'x$ tiene como varianza

$$Var(z_1) = \lambda_1, \tag{2.60}$$

siendo λ_1 el mayor autovalor de la matriz de covarianzas Σ y v_1 su autovector asociado de norma uno ($v_1'v_1 = 1$).

La segunda componente principal de X se define como z_2 que cumple

$$z_2 = v_2'x = v_{12}x_1 + ... + v_{d2}x_d \quad \text{con} \quad v_2 = (v_{12}, ..., v_{d2})' \in \mathcal{R}^d,$$

$$Var(z_2) = \text{máx}\{var(v'x) : v \in \mathcal{R}^d, v'v = 1, v'v_1 = 0\}. \tag{2.61}$$

Esta segunda componente es la combinación lineal de X formada por el vector unitario ortogonal a v_1 de mayor varianza. Se puede demostrar que la ortogonalidad entre los vectores v_1 y v_2

es equivalente a la incorrelación entre las componentes z_1 y z_2. En el caso de esta segunda componente principal, de forma análoga a la primera, la $Var(z_2) = \lambda_2$, siendo λ_2 el segundo mayor autovalor de la matriz de covarianzas Σ y v_2 su autovector asociado.

En general, las d componentes principales de x se definen como las variables aleatorias $z_1, ..., z_d$ que cumplen

$$z_1 = v_1'x, ..., z_d = v_d'x, \qquad v_1, ..., v_d \in \mathcal{R}^d \tag{2.62}$$

$$\begin{aligned} Var(z_1) &= \text{máx}\{Var(v'x) : v \in \mathcal{R}^d, v'v = 1\} \\ Var(z_2) &= \text{máx}\{Var(v'x) : v \in \mathcal{R}^d, v'v = 1, v_1'v = 0\} \\ &\vdots \quad \vdots \\ Var(z_j) &= \text{máx}\{Var(v'x) : v \in \mathcal{R}^d, v'v = 1, v_1'v = 0, ..., v_{j-1}'v = 0\} \\ &\vdots \quad \vdots \\ Var(z_d) &= \text{máx}\{Var(v'x) : v \in \mathcal{R}^d, v'v = 1, v_1'v = 0, ..., v_{d-1}'v = 0\} \end{aligned} \tag{2.63}$$

En este caso, también se cumple que

$$Var(z_j) = \lambda_j, j \in \{1, ..., d\}, \tag{2.64}$$

con $\lambda_1 \geq ... \geq \lambda_d \geq 0$ los d autovalores ordenados de la matriz de covarianzas Σ y $v_1, ..., v_d$ sus autovectores asociados normalizados. La covarianza de estos z_j es $Cov(z_j, z_k) = 0$ si $j \neq k$. Entonces,

$$z = V'x, \tag{2.65}$$

siendo $z = (z_1, ..., z_d)'$ y $V = (v_1, ..., v_d)$ la matriz cuyas columnas son los autovectores de Σ, por lo que

$$Cov(z, z) = V'\Sigma V. \tag{2.66}$$

Como la matriz de covarianzas de las componentes principales resulta diagonal con los autovalores de Σ como valores, el problema de componentes principales se reduce a la diagonalización de la matriz de covarianzas del vector aleatorio x.

La proporción de variabilidad explicada por las r primeras componentres principales viene dada por

$$\frac{\lambda_1 + ... + \lambda_r}{\lambda_1 + ... + \lambda_r + \lambda_{r+1} + ... + \lambda_d}. \tag{2.67}$$

Aquí nos encontramos con un problema: decidir entre la simplificación derivada de la reducción de la dimensión y la pérdida de información de la variabilidad no explicada. Para solucionar este problema existen, entre otros, tres criterios de decisión:

- Criterio de varianza explicada. Se utiliza el número de componentes principales que expliquen conjuntamente una proporción de varianza establecida, normalmente el 90 o 95 %.

- Gráfico de sedimentación. Se representa en un gráfico los valores de $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d$ en orden decreciente. Entonces, se busca un 'codo' en el gráfico, considerándose como codo el punto a partir del cual los valores son mucho más pequeños que los anteriores.
- Retener un número preestablecido de componentes principales. Habitualmente se escogen dos ya que se pueden representar graficamente.

Pero también hay que tener en cuenta que las componentes principales son sensibles ante cambios de escala. Si por ejemplo, se aumenta la escala de una de las variables originales de x, ésta verá incrementada su varianza y por lo tanto también su aportación a la variabilidad total, con lo que la primera componente principal va a tender a esta variable. Aunque si se cambia la escala en al misma proporción de todas las variables, el resultado de componentes principales no varía.

Pero este problema se puede solventar de dos formas: bien midiendo todas las variables en la misma escala o calculando las componentes principales con las variables estandarizadas. En este último caso se trabajaría con la matriz de correlaciones en vez de con la de covarianzas.

Las componentes principales se interpretan en función de si las variables originales están correlacionadas con ellas o no y, en caso de estar correlacionadas, si la correlación es positiva o negativa.

2.4.2. Mínimos cuadrados parciales

La regresión por mínimos cuadrados parciales es un modelo no lineal que encuentra las componentes ortogonales que forman una matriz Z, que continen las componentes z_m obtenidas con la siguiente restricción ([30]):

$$\text{máx}(cov(y, z_m)). \tag{2.68}$$

La regresión PLS se puede escribir matricialmente de la siguiente forma:

$$y = Zc' + \varepsilon, \tag{2.69}$$

siendo ε el vector de residuos y c el vector de coeficientes de las componentes. $T = XW^*$, por lo que podemos escribir

$$y = XW^{*z}c + \varepsilon, \tag{2.70}$$

siendo W^* la matriz de coeficientes de las variables x_j en cada componente z_m. Por otra parte, siendo $B = W^{*z}c$, podemos escribir

$$y = XB + \varepsilon, \tag{2.71}$$

que equivale a

$$y_i = \sum_{m=1}^{M} (c_m w^*_{1m} x_{i1} + ... + c_m w^*_{pm} x_{ip}) + \varepsilon_i, \tag{2.72}$$

con M el número de componentes retenidas en el modelo final que, en general, es muy inferior al rango de X, y p es igual al número de variables contenidas en la matriz de X. Entonces,

$$y_i = \sum_{j=1}^{p} b_j x_{ij} + \varepsilon_i \tag{2.73}$$

$$\text{donde} \quad b_j = \sum_{m=1}^{M} c_m w_{jm}^*, \quad j \in \{1, ..., p\}. \tag{2.74}$$

w_{jm}^* refleja la relación entre el vector y y las variables x_j a través de las componentes z_m. Éstos serán los parámetros que se utilizarán en los análisis y para comparar las propiedades de los diferentes modelos.

A continuación, presentaremos un algoritmo propuesto por [16] para realizar la estimación por mínimos cuadrados parciales.

Igual que el método de componentes principales, mínimos cuadrados parciales (PLS) no es invariante ante cambios de escala, por lo que asumimos que cada x_j está estandarizado para tener media 0 y varianza 1. PLS comienza computando $w_{1m}^* =< x_j, y >$ para cada j. Entonces construimos $z_1 = \sum_j \hat{w}_{1m}^* x_j$, que es la primera dirección mínimo cuadrada parcial. Vemos que en cada z_m los inputs son ponderados de acuerdo a su efecto univariante sobre la variable respuesta y. Se realiza la regresión de y sobre z_1 obteniendo el coeficiente $\hat{\theta}_1$, y ortogonalizamos $x_1, ..., x_p$ con respecto a z_1. Continuamos este proceso hasta que se obtengan $M \leq p$ direcciones.

Igual que ocurría en componentes principales, si construimos $M = p$ direcciones, obtendremos la solución equivalente al método de mínimos cuadrados.

En el siguiente algoritmo se detalla el proceso necesario para aplicar este método.

1. Estandarizar cada x_j para que tenga media 0 y varianza 1. Fijar $\hat{y}^{(0)} = \overline{y}_1$, y $x_j^{(0)} = x_j$, $j = 1, ..., p$.
2. Para $m = 1, 2, ..., p$:
   a) $z_m = \sum_{j=1}^{p} \hat{w}_{mj}^* x_j^{(m-1)}$, donde $\hat{w}_{mj}^* =< x_j^{(m-1)}, y >$
   b) $\hat{c}_m =< z_m, y > / < z_m, z_m >$
   c) $\hat{y}^{(m)} = \hat{y}^{(m-1)} + \hat{c}_m z_m$
   d) Ortogonalizar cada $x_j^{(m-1)}$ con respecto a
   $z_m : x_j^{(m)} = x_j^{(m-1)} - \left[< z_m, x_j^{(m-1)} > / < z_m, z_m >\right] z_m$, $j = 1, 2, ..., p$
3. Obtener la secuencia de vectores ajustados $\{\hat{y}^{(m)}\}_1^p$. Debido a que $\{z_\rho\}_1^m$ son lineales en la x_j original, $\hat{y}^{(m)} = X\hat{\beta}^{pls}(m)$. Estos coeficientes lineales pueden ser recuperados de la secuencia de transformaciones PLS.

Como se utiliza la respuesta y para construir las direcciones, las soluciones que obtenemos son funciones no lineales de y. En este método se buscan las direcciones que tienen varianzas y correlaciones altas con y, a diferencia del método de componentes principales que solamente busca las direcciones con altas varianzas.

El problema de optimización que resolvemos en mínimos cuadrados parciales para la m-ésima dirección es

$$\begin{aligned} \max_{\alpha} \quad & Corr^2(y, X_\alpha)Var(X_\alpha) \\ \text{s.a.} \quad & ||\alpha|| = 1, \hat{w}_l^{'*} S\alpha = 0, l = 1, ..., m-1 \end{aligned} \tag{2.75}$$

siendo S la matriz de varianzas-covarianzas de x_j.

En el caso de la regresión logística por mínimos cuadrados parciales [30], construimos en cada etapa la regresión de y sobre las componentes $z_1, ..., z_m$. La ecuación de regresión logística PLS se obtiene expresando estas ecuaciones en función de las variables x_j. Así, para una respuesta y, en función de la probabilidad π de que $Y = 1$, obtenemos

$$\begin{aligned} \widehat{Logit(\pi)} &= c_1 z_1 + ... + c_m z_m \\ &= c_1 X w *_1 + ... + c_m X w *_m \\ &= XB, \end{aligned} \tag{2.76}$$

con $m \in 1, ..., M$, siendo M el número de componentes retenidas en el modelo final, y $M \leq rango(X)$. En la regresión logística PLS, las componentes z_m se construyen de forma iterativa a partir de las regresiones logísticas individuales $Logit(\mathbb{P}(Y = 1|x_j)) = \beta_{0j} + \beta_j x_j$.

Entonces, el parámetro de interés que expresa la relación entre $Logit(\pi)$ y X es β. En la práctica, las componentes de la regresión logística PLS se obtienen de forma iterativa utilizando el algoritmo NIPALS (Mínimos cuadrados parciales iterativos no lineales) [38]. Este algoritmo se presenta a continuación.

Comienza con las matrices X e Y opcionalmente transformadas, reescaladas y centradas, y se procede de la siguiente forma:

1. Crear un vector u, normalmente una de las columnas de Y. Cuando solamente hay una columan, $u = y$.
2. Los pesos de X: $w = X'u/u'u$. Ahora podemos transformar (a elección del investigador) w en $\|w\| = 1.0$.
3. Calcular los scores de X t: $t = Xw$.
4. Los pesos de Y, c: $c = Y't/t't$.
5. Actualizar el conjunto de scores de Y u: $u = Yc/c'c$.
6. Comprobar la convergencia de t mediante $\|t_{old} - t_{new}\|/\|t_{new}\| < \varepsilon$, donde ε es una cantidad muy pequeña. Si no se da la covergencia, volver al paso 2, en otro caso, continuar con el paso 7. Si Y es solamente una variable, el proceso converge en una única iteración, por lo que se procede directamente al paso 7.
7. Eliminar la componente actual de X e Y, utiliza esas matriz modificadas X e Y en la siguiente componente. La modificación de Y es opcional ya que el resultado va a ser el mismo.
   $p = X't/(t't)$

$X = X - tp'$
$Y = Y - tc'$

8. Volvemos al paso 1 con la siguiente componente hasta que la validación cruzada indique que no hay más variables significativas.

2.5. Modelos aditivos generalizados en alta dimensión

En esta sección expondremos algunas de las extensiones existentes de los modelos aditivos generalizados que se pueden utilizar en el caso en el que $p \gg N$.

2.5.1. Modelos aditivos con penalización sparsity-smoothness

Considerando el modelo aditivo en alta dimensión con respuesta continua $Y \in \mathbb{R}^n$ y p covariables $x^{(1)}, ..., x^{(p)} \in \mathbb{R}^n$,

$$Y_i = \alpha + \sum_{j=1}^{p} f_j(x_i^{(j)}) + \varepsilon_i, \quad i = 1, .., n, \tag{2.77}$$

donde α es el intercepto, ε_i son variables aleatorias i.i.d. con media cero y $f_j : \mathbb{R} \to \mathbb{R}$ son funciones suavizadoras univariantes. Asumimos también que todas las funciones f_j están centradas, es decir,

$$\sum_{i=1}^{n} f_j(x_i^{(j)}) = 0, \tag{2.78}$$

para $j = 1, ..., p$. Consideramos el caso de diseño fijo, es decir, las covariables $x^{(1)}, ..., x^{(p)}$ no son aleatorias. Permitiéndonos cierto abuso del lenguaje, denotamos por f_j el vector n-dimensional $(f_j(x_1^{(j)}), ..., f_j(x_n^{(j)}))$. Para un vector $f \in \mathbb{R}^n$, definimos $\|f\|_n^2 = \frac{1}{2}\sum_{i=1}^{n} f_i^2$.

Si utilizamos un número alto de funciones base, lo cual es necesario para capturar algunas funciones complejas, se producirán dos fenómenos que debemos penalizar: *sparsity* y *roughness*. Esta situación también provoca que el estimador resultante produzca funciones estimadas demasiado móviles si la verdadera función es muy suavizada. Por eso, se necesitan aplicar restricciones a las funciones estimadas. Para obtener funciones estimadas suficientemente suavizadas pero no demasiado suavizadas, en [28] se propone la penalización *sparsity-smoothness*

$$J(f_j) = \lambda_1 \sqrt{\|f_j\|_n^2 + \lambda_2 I^2(f_j)}, \tag{2.79}$$

donde

$$I^2(f_j) = \int (f_j''(x))^2 dx \tag{2.80}$$

mide la suavidad de f_j y los parámetros $\lambda_1, \lambda_2 \geq 0$ controlan el nivel de penalización.

El estimador se obtiene del problema de mínimos cuadrados penalizados siguiente:

$$\hat{f}_1, ..., \hat{f}_p = \underset{f_1,...,f_p \in \mathcal{F}}{\arg\min} \left\| Y - \sum_{j=1}^{p} f_j \right\|_n^2 + \sum_{j=1}^{p} J(f_j), \tag{2.81}$$

donde $\mathcal{F}$ es una clase de funciones y $Y = (Y_1, ..., Y_n)^T$ es el vector respuesta. Se asume el mismo nivel de regularización para cada función f_j.

En [28] se demuestra que si $a, b \in \mathbb{R}$ tal que $a < \text{mín}_{i,j}\{x_i^{(j)}\}$ y $b > \text{máx}_{i,j}\{x_i^{(j)}\}$. Y sea $\mathcal{F}$ el espacio de funciones continuas y diferenciables en $[a, b]$ y asumiendo que existe el mínimo $\hat{f}_j \in \mathcal{F}$ de (2.81). Entonces, las $\hat{f}$ son splines cúbicos naturales con nodos en $x_i^{(j)}, i = 1, ..., n$. En consecuencia, podemos restringirnos al espacio finito-dimensional de splines cúbicos naturales en lugar de considerar el espacio infinito-dimensional de funciones continuas y diferenciables.

A partir de aquí, para cada función f_j utilizaremos una parametrización de B-spline cúbico con un número razonable de nodos o funciones base. Se suelen utilizar $k - 4 \asymp \sqrt{n}$ nodos interiores, que se sitúan en los cuantiles empíricos de $x^{(j)}$. Entonces,

$$f_j(x) = \sum_{k=1}^{K} \beta_{j,k} b_{j,k}(x), \tag{2.82}$$

donde $b_{j,k} : \mathbb{R} \to \mathbb{R}$ son los B-splines y $\beta_{j,k} = (\beta_{j,1}, ..., \beta_{j,K})^T \in \mathbb{R}^K$ es el vector de parámetros correspondiente a f_j. Basándonos en las funciones base, podemos construir una matriz de diseño n x pK $B = [B_1|B_2|...|B_p]$, donde B_j es la matriz de diseño n x K de la base B-spline del j-ésimo predictor, es decir, $B_{j,il} = b_{j,l}(x_i^{(j)})$.

Entonces, para funciones continuas diferenciables, el problema de optimización (2.81) puede ser reformulado como

$$\hat{\beta} = \underset{\beta=(\beta_1,...,\beta_p)}{\arg\min} \|Y - B\beta\|_n^2 + \lambda_1 \sum_{j=1}^{p} \sqrt{\frac{1}{n}\beta_j^T B_j^T B_j \beta_j + \lambda_2 \beta_j^T \Omega_j \beta_j}, \tag{2.83}$$

donde la matriz KxK Ω contiene los productos interiores de las segundas derivadas de las funciones base de los B-splines, esto es,

$$\Omega_{j,kl} = \int b''_{j,k}(x) b''_{j,l}(x) dx, \tag{2.84}$$

para $k, l \in \{1, ..., K\}$. Entonces, (2.83) se puede reescribir como un problema lasso agrupado general

$$\hat{\beta} = \underset{\beta=(\beta_1,...,\beta_p)}{\arg\min} \|Y - B\beta\|_n^2 + \lambda_1 \sum_{j=1}^{p} \sqrt{\beta_j^T M_j \beta_j}, \tag{2.85}$$

donde $M_j = \frac{1}{n} B_j^T B_j + \lambda_2 \Omega_j$. Mediante la descomposición de Cholesky, $M_j = R_j^T R_j$ para alguna matriz KxK cuadrática R_j y definiendo $\tilde{\beta}_j = R_j \beta_j, \tilde{B}_j = B_j R_j^{-1}$, se reduce (2.85) a

$$\hat{\tilde{\beta}} = \underset{\tilde{\beta}=(\tilde{\beta}_1,...,\tilde{\beta}_p)}{\arg\min} \|Y - \tilde{B}\tilde{\beta}\|_n^2 + \lambda_1 \sum_{j=1}^{p} \|\tilde{\beta}_j\|, \tag{2.86}$$

donde $\|\tilde{\beta}\| = \sqrt{K}\|\tilde{\beta}_j\|_K$ es la norma euclídea en $\mathbb{R}^K$. Éste es un problema lasso agrupado ordinario para cualquier λ_2 fijo, por lo que la existencia de solución está garantizada. Para λ_1 suficientemente grande, algunos de los coeficientes $\beta_j \in \mathbb{R}^K$ se reducirán a cero, por lo que su correspondiente función también se estimará como cero. Además, existe un valor $\lambda_{1,max} < \infty$ tal que $\hat{\tilde{\beta}}_1 = ... = \hat{\tilde{\beta}}_p = 0$ para $\lambda_1 \geq \lambda_{1,max}$.

2.5.2. SpAM: Sparse Additive Models

Los modelos SpAM ([32]) extienden las ventajas de los modelos lineales *sparse* a los modelos aditivos. El modelo base del que parte es del modelo aditivo generalizado:

$$Y_i = \sum_{j=1}^{p} f_j(X_{ij}) + \varepsilon_i, \tag{2.87}$$

pero le imponemos una restricción de *sparsity* al conjunto de índices $\{j : f_j \neq 0\}$ de las funciones f_j que no son cero.

Supongamos que tenemos la muestra $(X_1, Y_1), ..., (X_n, Y_n)$ donde $X_i = (X_{i1}, ..., X_{ij}, ..., X_{ip})^T \in [0,1]^p$ y

$$Y_i = m(X_i) + \varepsilon_i, \tag{2.88}$$

con $\varepsilon_i \sim N(0, \sigma^2)$ y

$$m(x) = \sum_{j=1}^{p} f_j(x_j). \tag{2.89}$$

Denotando la distribución conjunta de (X_i, Y_i) por P. Para una función f en $[0,1]$, su norma $L_2(P)$ es de la forma

$$\|f\| = \sqrt{\int_0^1 f^2(x) dP(x)} = \sqrt{\mathbb{E}(f)^2}. \tag{2.90}$$

A continuación, comenzaremos formulando el problema de optimización a nivel poblacional. Entonces, añadiremos estimaciones suavizadas para finalmente terminar con el algoritmo back-fitting sparse.

En primer lugar, por simplicidad, asumiremos que $\mathbb{E}(Y_i) = 0$. El problema de optimización del modelo aditivo estandar en $L_2(P)$ es

$$\min_{f_j \in \mathcal{H}_j, 1 \leq j \leq p} \mathbb{E}\left(Y - \sum_{j=1}^{p} f_j(X_j)\right)^2, \tag{2.91}$$

donde la esperanza es tomada con respecto a X y el ruido ε. Ahora modificamos el problema

para introducir un parámetro de escala a cada función e imponemos restricciones adicionales:

$$\begin{aligned} \min_{\beta\in\mathbb{R}p, g_j\in\mathcal{H}_j} \quad & \mathbb{E}(Y-\sum_{j=1}^{p}\beta_j g_j(X_j))^2 \\ \text{sujeto a:} \quad & \sum_{j=1}^{p}|\beta_j|\leq L, \\ & \mathbb{E}(g_j^2)=1, j=1,...,p. \end{aligned} \tag{2.92}$$

La restricción de que β se encuentra en la bola en $L_1\{\beta : \|\beta\|_1 \leq L\}$ incrementa la sparsity de la β estimada. El problema de optimización (2.92) se puede reescribir de forma equivalente como

$$\begin{aligned} \min_{f_j\in\mathcal{H}_j} \quad & \mathbb{E}(Y-\sum_{j=1}^{p} f_j(X_j))^2 \\ \text{sujeto a} \quad & \sum_{j=1}^{p}\sqrt{\mathbb{E}(f_j^2(X_j))}\leq L, \end{aligned} \tag{2.93}$$

que a su vez, se puede escribir en forma lagrangiana:

$$L(f,\lambda)=\frac{1}{2}\mathbb{E}(Y-\sum_{j=1}^{p}f_j(X_j))^2+\lambda\sum_{j=1}^{p}\sqrt{\mathbb{E}(f_j^2(X_j))}, \tag{2.94}$$

cuya solución es

$$f_j=\left[1-\frac{\lambda}{\sqrt{\mathbb{E}(P_j^2)}}\right]_+ P_j \qquad \text{a.s.}, \tag{2.95}$$

donde $[\cdot]_+$ denota la parte positiva y $P_j=\mathbb{E}[R_j|X_j]$ denota la proyección de los residuos $R_j = Y-\sum_{k\neq j}f_k(X_k)$ sobre $\mathcal{H}_j$, como se demuestra en [32].

Una vez obtenida la solución a nivel poblacional, para obtener la versión muestral insertaremos estimaciones muestrales en el algoritmo poblacional, como sucede en el algortimo backfitting estándar. Entonces, estimaremos la proyección $P_j=\mathbb{E}(R_j|X_j)$ mediante residuos suavizados:

$$\hat{P}_j=S_jR_j \tag{2.96}$$

donde S_j es un suavizador lineal, como por ejemplo el suavizador lineal local o el suavizador kernel. Siendo

$$\hat{s}_j=\frac{1}{\sqrt{n}}\|\hat{P}_j\|=\sqrt{\mu(\hat{P}_j)}, \tag{2.97}$$

la estimación de $\sqrt{\mathbb{E}(P_j^2)}$. Sustituyendo esto en la ecuación (2.95), obtenemos el algoritmo backfitting SpAM.

Algoritmo Backfitting SpAM:

Input: Datos (X_i, y_i), λ parámetro de regularización.
Inicializar: $\hat{f}_j=0$, para $j=1,...,p$.

Iteración hasta convergencia:

Para cada $j = 1, ..., p$:

1. Calcular los residuos: $R_j = Y - \sum_{k \neq j} \hat{f}_k(X_k)$.
2. Estimar $P_j = \mathbb{E}[R_j|X_j]$ mediante el suavizador $\hat{P}_j = S_j R_j$.
3. Estimar $\hat{s}_j^2 = \frac{1}{n} \sum_{i=1}^n \hat{P}_j^2(i)$.
4. $\hat{f}_j = [1 - \lambda/\hat{s}_j]_+ \hat{P}_j$.
5. $\hat{f}_j \leftarrow \hat{f}_j - mean(\hat{f}_j)$.

Output: funciones componentes $\hat{f}_j$ y estimador $\hat{m}(X_i) = \sum_j \hat{f}_j(X_{ij})$.

Además, este modelo se puede extender a la regresión logística no paramétrica para clasificación. El modelo aditivo logístico es de la forma:

$$\mathbb{P}(Y = 1|X) \equiv p(X; f) = \frac{exp\left(\sum_{j=1}^p f_(X_j)\right)}{1 + exp\left(\sum_{j=1}^p f_j(X_j)\right)}, \tag{2.98}$$

donde $Y \in \{0, 1\}$ y la log-verosimilitud poblacional es

$$l(f) = \mathbb{E}[Y f(X) - log(1 + expf(X))]. \tag{2.99}$$

Renombrándolo en el algoritmo de scoring local para modelos aditivos generalizados para el caso logístico, utilizaremos el proceso backfitting con el método de Newton. En este caso, se computa iterativamente la respuesta transformada para la estimación de f_0

$$Z_i = f_0(X_i) + \frac{Y_i - p(X_i; f_0)}{p(X_i; f_0)(1 - p(X_i; f_0))} \tag{2.100}$$

y pesos $w(X_i) = p(X_i; f_0)(1 - p(X_i; f_0))$, y lleva a cabo un backfitting ponderado de (Z, X) con pesos w. El suavizador ponderado es

$$\hat{P}_j = \frac{S_j(wR_j)}{S_j w}. \tag{2.101}$$

Para incorporar la penalización de sparsity, expresamos la función anterior en su forma lagrangiana

$$L(f, \lambda) = \mathbb{E}[\log(1 + e^{f(X)}) - Y f(X)] + \lambda \left(\sum_{j=1}^p \sqrt{\mathbb{E}(f_j^2(X_j))} - L \right) \tag{2.102}$$

y la condición estacionaria para f_j es $\mathbb{E}(p - Y|X_j) + \lambda v_j = 0$, donde v_j es un elemento del subgradiente $\partial\sqrt{\mathbb{E}(f_j^2(X_j))}$. Pero esta condición es no lineal en f, por lo que linealizamos el gradiente de la log-verosimilitud en f_0. Esto nos lleva a la condición linealizada $\mathbb{E}[w(X)(f(X) - Z)|X_j] + \lambda v_j = 0$. Entonces $\mathbb{E}(f_2^2) \neq 0$, lo que implica

$$\left(\mathbb{E}(w|X_j) + \frac{\lambda}{\sqrt{\mathbb{E}(f_j^2)}} \right) f_j(X_j) = \mathbb{E}(wR_j|X_j). \tag{2.103}$$

En el caso de una muestra finita, en términos de la matriz de suavizado S_j, esto se convierte en

$$f_j = \frac{S_j(wR_j)}{S_j w + \lambda/\sqrt{\mathbb{E}(f_j^2)}}. \tag{2.104}$$

Si $\|S_j(wR_j)\| < \lambda$, entonces $f_j = 0$. Por otro lado, esto implica que las ecuaciones no lineales para f_j no pueden ser resueltas explicitamente, por lo que [32] propone iterar hasta convergencia:

$$f_j \leftarrow \frac{S_j(wR_j)}{S_j w + \lambda\sqrt{n}/\|f_j\|}. \tag{2.105}$$

Cuando $\lambda = 0$ esto se convierte en la ecuación (2.101).

Para seleccionar el parámetro de suavizado λ utilizaremos los métodos habituales como, por ejemplo, C_p o GCV.

2.5.3. GAMSEL

Con datos (y_i, x_i) para $i = 1, ..., n$. Representamos

$$f_j(x) = \alpha_j x_j + u_j(x_j)^T \beta_j, \tag{2.106}$$

donde u_j es un vector de m_j bases de funciones. $U_j \in \mathbb{R}^{nxm_j}$ es la matriz de evaluación de estas funciones, y asumimos que U_j tiene columnas ortonormales, sin pérdida de generalidad.

El método GAMSEL ([7]) estima f_j resolviendo el siguiente problema de optimización convexo

$$\begin{aligned}\min_{\alpha_0,\{\alpha_j\},\{\beta_j\}} \frac{1}{2}\left\| y - \alpha_0 - \sum_{j=1}^{p}\alpha_j x_j - \sum_{j=1}^{p} U_j\beta_j \right\|_2^2 + \lambda \underbrace{\sum_{j=1}^{p}(\gamma|\alpha_k| + (1-\gamma)\|\beta_j\|_{D*_j})}_{\text{penalización de selección}} \\ + \frac{1}{2} \underbrace{\sum_{j=1}^{p}\psi_j\beta_j^T D_j\beta_j}_{\text{penalización } \textit{end-of-path}},\end{aligned} \tag{2.107}$$

donde $\|\beta_j\|_{D+_j} = \sqrt{\beta_j^T D *_j \beta_j}$. Por simplicidad de notación, tanto y como x_j son vectores de dimensión n.

En primer lugar nos centraremos en la penalización *end-of-path*, que es lo único que permanece activo cuando $\lambda = 0$. El multiplicador ψ_j para cada término se escoge para que el ajuste de solamente ese término resulte en unos grados de libertad pre-especificados. Entonces, cuando $\lambda = 0$, ajustamos un modelo aditivo generalizado con grados de libertad pre-especificados para cada término.

La penalización de selección es más compleja, y consiste en una mixtura de penalizaciones en L_1 y L_2 para cada término. Éste toma la forma de una superposición de penalizaciones de lasso agrupado, que tiene el efecto de inducir sparsity en el modelo ajustado. El término $\|\beta_j\|_{D*_j}$ es una penalización del lasso agrupado. La superposición anteriormente mencionada se refiere a el hecho de que cada x_j tiene un par de coeficientes lineales, uno representado en

$\|\beta_j\|_{D*_j}$, y el otro en $|\alpha_j|$. Aquí la matriz $D*_j$ es idéntica a D_j, exceptuando que el cero en la primera posición se reemplaza por un 1, es decir, se penaliza el término lineal, siendo D la matriz de penalizaciones. El parámetro γ se encuentra entre 0 y 1, y permite que se opte por términos lineales (γ pequeño) en lugar de términos no lineales, o viceversa. Debido a la particular estructura de esta penalización, hay tres posibilidades para cada predictor.

- Zero ($\alpha = 0, \beta_j \equiv 0$). Para valores altos de λ, el término de penalización puede dominar el término de ajuste, el cual resulta de minimizar teniendo $\alpha_j = 0$ y $\beta_j \equiv 0$. Esto se corresponde con el caso de que $f_j(x) = 0$.
- Lineal ($\alpha \neq 0, \beta_j \equiv 0$). Para valores moderados de λ y $\gamma > 0$ suficientemente pequeños, el resultado de minimizar puede tener $\alpha_j \neq 0$ y $\beta_j \equiv 0$. Esto se corresponde con el caso en el que $f_j(x) = \alpha_j x$, que se estima para que la función de x sea estrictamente lineal.
- No lineal ($\beta_j \neq 0$). Para valores pequeños de λ y/o valores grandes de γ, el resultado de minimizar puede ser $\beta_j \neq 0$. Esto se corresponde con ajustar una curva de la forma $f_j(x) = \alpha_j x + U_j \beta_j$ para el j-ésimo predictor.

3. Funciones de R para modelos de regresión en alta dimensión

3.1. Funciones de R para la selección de subconjuntos

Para el caso de la selección de subconjuntos de covariables, utilizaremos la función de R `regsubsets()` de la librería `leaps` ([35]). Esta función puede utilizar el método de selección forward-stepwise, el de backward-stepwise o el método de selección del mejor subconjunto. Los principales elementos que nos ofrece esta función son:

- `nvmax`: es el tamaño máximo de los subconjuntos a examinar. Por defecto es igual a 8.
- `intercept`: nos da la opción de examinar los modelos con o sin intercepto. Por defecto incorpora el intercepto.
- `method`: nos da a escoger el método que queremos utilizar: `"exhaustive"`, búsqueda exhaustiva (se corresponde con el método de selección del mejor subconjunto), es la opción por defecto; `"forward"`, selección forward; `"backward"`, selección backward; o `"seqrep"`, reemplazamiento secuencial para la búsqueda.
- `really.big`: esta opción sirve para indicar si estamos en alta dimensión o no. Considera alta dimensión a partir de 50 covariables.

Esta función no devuelve ninguna salida directamente, sino que crea un objeto de la clase `regsubsets`. Para poder ver lo que se encuentra en el objeto debemos utilizar la función `summary()`. Los principales elementos que nos devuelve esta función son:

- `which`: es una matriz lógica que indica que elementos se encuentran en cada modelo.
- `rsq`: R^2 para cada modelo.
- `rss`: RSS (suma de residuos al cuadrado) para cada modelo.
- `adjr2`: R^2 ajustado para cada modelo.
- `cp`: Cp de Mallows.
- `bic`: Schwartz's information criterion, BIC
- `outmat`: es una versión de `which` en formato para imprimir.
- `obj`: es una copia del objeto `regsubsets`.

Para poder utilizar el método de selección del mejor subconjunto en R, utilizaremos la función `regsubsets` de la librería `leaps`. En el argumento `method` que nos ofrece esta función debemos especificar la opción "`exhaustive`".

Para el caso de selección forward-stepwise, deberemos especificar la opción "`forward`"en el elemento `method` y, para el caso del método de selección backward-stepwise, la opción a escoger será "`backward`".

3.2. Funciones de R para los métodos de regularización

En este apartado describiremos las funciones de R que utilizaremos para aplicar los modelos de regularización.

3.2.1. Regresión ridge y lasso

Tanto para la regresión ridge como para la regresión lasso utilizaremos la función `glmnet()` de la librería `glmnet` [15]. Esta función ajusta modelos lineales generalizados mediante máxima verosimilitud penalizada, utilizando el método lasso o penalización elasticnet. Puede ajustar modelos de regresión lineales, logísticos, multinomiales, de poisson y de Cox. Esta función resuelve en un grid de valores de λ el problema

$$\min_{\beta_0,\beta} \frac{1}{N} \sum_{i=1}^{N} \left\{ w_i I(y_i, \beta_0 + \beta^T x_i) + \lambda[(1-\alpha)\|\beta\|_2^2/2 + \alpha\|\beta\|_1] \right\} \tag{3.1}$$

donde $I(y, \eta)$ es la contribución negativa de la log-verosimilitud por cada observación i.

La penalización elasticnet se controla mediante $\alpha \in [0, 1]$, siendo $\alpha = 1$ el método lasso y $\alpha = 0$ la regresión ridge.

Esta función tiene numerosos elementos, pero los principales son:

- `x`: es la matriz de inputs de dimensión "número de observaciones"x "número de variables".
- `y`: es la variable respuesta. Esta debe ser cuantitativa para `family=`"`gaussian`"y para `family=`"`poisson`". Para `family=`"`binomial`"debe ser o una variable factorial con dos niveles o una matriz con dos columnas. Para `family=`"`multinomial`"debe ser una variable factorial con 2 o más factores, o una matriz con 2 o más columnas. En caso de ser `family=`"`cox`", debe ser una matriz de dos columnas con una columna llamada "timez otra "status".
- `family`: nos permite escoger que distribución sigue la variable respuesta. Ésta puede ser: "`gaussian`", "`binomial`", "`multinomial`", "`poisson`"o "`cox`".
- `alpha`: es el parámetro que controla la penalización elasticnet. Por defecto $\alpha = 1$ (método lasso).
- `nlambda`: es el número de valores de λ que queremos en el grid en el que se realiza el ajuste. Por defecto son 100.
- `lambda`: aquí especificamos si queremos una secuencia de valores de λ concreta.
- `intercept`: escogemos si queremos realizar el ajuste con o sin intercepto.
- `dfmax`: limitamos el número máximo de variables en el modelo.

Esta función devuelve un objeto de tipo "`glmnet`","`*`", donde "`*`"depende de la distribución de la variable respuesta que hayamos escogido. Los principales elementos de este objeto son:

- `call`: el comando con el que llamamos a la función y sus opciones.
- `a0`: la secuencia de intercepto de longitud `length(lambda)`.
- `beta`: devuelve una matriz `nvars` x `length(lambda)` de coeficientes, exceptuando cuando la distribución de `y` es multinomial o normal multivariante que devuelve una lista de matrices, una para cada clase.
- `lambda`: devuelve la secuencia de valores de λ utilizada.
- `dev.ratio`: la fracción de deviance explicada.
- `df`: el número de coeficientes distintos de cero para cada valor de `lambda`.

Como ya hemos comentado, esta función ajusta el modelo para un grid de valores de λ, pero no nos da el λ óptimo que debemos escoger. Para eso, en la misma librería, disponemos de la función `cv.glmnet()`, que nos da el mejor λ según el criterio de validación cruzada. Los principales elementos que podemos manipular de esta función son similares a los de la función `glmnet()`, a excepción de dos:

- `nfolds`: es el número de iteraciones (por defecto 10), que tienen que ser como mínimo 3.
- `type.measure`: sirve para escoger el tipo de medida de la desviación de la media ajustada a la respuesta. Por defecto es `type.measure="deviance"`, que en caso de ser un modelo gaussiano es el error cuadrático medio.

Lo que varía es la salida que nos ofrece, cuyos elementos más interesantes son:

- `lambda`: los valores de `lambda` utilizados para el ajuste.
- `cvm`: es el error medio de la validación cruzada.
- `cvsd`: la estimación del error estándar de `cvm`.
- `nzero`: número de coeficientes distintos de cero para cada `lambda`.
- `lambda.min`: el valor de `lambda` que da el mínimo `cvm`.
- `lambda.1se`: máximo valor de `lambda` que da lugar a un error por debajo de un error típico del error mínimo.

3.2.2. Least Angle Regression

Para llevar a cabo la regresión Least Angle en R, utilizaremos dos funciones distintas, una para el caso de respuesta variable continua y otra para la variable binaria.

En el primer caso, la función que utilizaremos será la función `lars()` de la librería `lars` [19]. Los principales elementos de esta función son:

- `x`: la matriz de covariables.
- `y`: la variable respuesta.
- `type`: nos permite escoger entre cuatro métodos distintos de regresión: `"lasso"` (por defecto), `"lar"`, `"forward.stagewise"` y `"stepwise"`.

- `normalize`: si es `TRUE` (por defecto), estandariza cada variable para tener norma L2 unitaria.
- `intercept`: si es `TRUE` (por defecto), se incluye intercepto en el modelo pero no se penaliza, en otro caso no se incluye el intercepto.

Esta función deuelve un objeto `lars`, al cual le podemos aplicar, entre otras, las funciones `summary()`, `coef()`, etc, para analizar así los resultados obtenidos con la función. La función `lars()` solamente clasifica los distintos modelos mediante los criterios RSS (Suma de Residuos al Cuadrado) y el C_p de Mallow.

Para el caso de variable respuesta binaria, utilizaremos la función `dglars()` de la librería `dglars` [3], la cual deberemos instalar y cargar para poder utilizar. Esta función solamente la podremos utilizar en caso de que la variable respuesta siga una distribución binomial o de poisson. Esta función está diseñada para utilizar en el caso de que $p > n$.

Puede utilizar dos algoritmos distintos para realizar la estimación: método predictor-corrector y método cyclic coordinate descent. En nuestro caso utilizaremos el primero. Este consiste en dos fases: en la primera, llamada paso predictor, se realiza una aproximación de la solución del algoritmo; en la segunda, llamada paso corrector, se utiliza el algoritmo de Newton-Raphson para corregir la solución aproximada obtenida en la primera fase.

Los elementos más interesantes de esta función son:

- `formula`: un objeto de clase "formula": una descripción del modelo que queremos ajustar.
- `family`: una descripción de la distribución del error utilizado en el modelo. Esta puede ser "`binomial`"o "`poisson`".
- `control`: es una lista de parámetros de control. Esta lista tiene, entre otros, los siguientes elementos:
  - `algorithm`: debemos especificar el algoritmo que queremos utilizar para ajustar el modelo LAR. Las opciones son: "`pc`", para el método predictor-corrector, que es el que escoge por defecto, o "`ccd`", que es el método cyclic coordinate descent.
  - Otros elementos para ajustar lo máximo posible a nuestro caso el algoritmo a utilizar.

Esta fórmula nos devuelve un objeto "`dglars`", que es una lista con los siguientes componentes:

- `call`: la llamada que produce el objeto.
- `family`: descripción de la distribución del error utilizado en el modelo.
- `np`: el número de puntos de la solución.
- `beta`: la matriz $(p+1)*$`np` con la solución del ajuste.
- `dev`: el vector con la deviance correspondiente a los valores del parámetro de ajuste.
- `df`: la secuencia del número de coeficientes distintos de cero para cada valor del parámetro de ajuste.
- `g`: la secuencia de parámetros de ajuste utilizados para computar la solución.

Con la función `summary.dglars()` de la misma librería, podemos obtener el mejor modelo LAR según una medida de bondad del ajuste (GoF). La medida que utiliza esta función es:

$$Dev + K complexity, \tag{3.2}$$

donde *Dev* es la deviance residual, *complexity* es el término utilizado para medir la complejidad del ajuste del modelo y K es el término utilizado para ponderar la complejidad en la fórmula de GoF. Los elementos *complexity* y K son los que podemos escoger en la función `summary`, que nos ofrecen las siguientes opciones:

- `K = "BIC"` (por defecto) para utilizar el método BIC; y `k = "AIC"` para utilizar el método AIC.
- `complexity = "df"`, en este caso, la complejidad está definida como el número de coeficientes distintos de cero; y `complexity = "gdf"`, que representa los grados de libertad generalizados, útil en caso de regresión logística.

3.3. Funciones de R para los modelos de reducción dimensión

3.3.1. Regresión de componentes principales

Para llevar a cabo la regresión de componentes principales con datos de alta dimensión hemos utilizado la función `pcr()` de la librería `pls` [29]. Fue esta función la escogida ya que nos permite realizar el análisis de componentes principales con datos tanto continuos como discretos.

Los principales elementos que nos ofrece esta función son:

- `formula`: la fórmula del modelo.
- `ncomp`: número de componentes principales que se mantienen en el resultado.
- `scale.unit`: si es `TRUE` (opción por defecto), los datos son reescalados para tener varianza unitaria.
- `subset`: vector opcional indicando el subconjunto de observaciones utilizadas en el ajuste.
- `validation`: el tipo de validación que queremos utilizar. Las opciones son: `ÇV"`, validación cruzada (por defecto); `"LOO"` y validación cruzada leave-one-out.

Esta función nos devuelve una lista con los siguientes elementos:

- `validation`: si se utiliza validación, es su resultado.
- `ncomp`: número de componentes del modelo.
- `terms`: los términos del modelo.

3.3.2. Partial least squares

Para estimar el modelo de mínimos cuadrados parciales con variable respuesta continua, utilizaremos la función `plsr()` de la librería `pls` [29]. Esta función tiene los mismo elementos y nos devuelve lo mismo que la función `pcr()`, que utilizamos para el método de componentes principales.

En el caso del modelo mínimos cuadrados parciales con variable respuesta binaria, utilizaremos la función de R `plsRglm()` de la librería `plsRglm`. Los elementos más interesantes de esta función son:

- `nt`: número de componentes que se van a extraer.
- `modele`: es el nombre del modelos que se va a ajustar, las opciones son: `"pls"` (por defecto), `"pls-glm-Gamma"`, `"pls-glm-gaussian"`, `"pls-glm-inverse.gaussian"`, `"pls-glm-logistic"`, `"pls-glm-poisson"` y `"pls-glm-polr"`.
- `MClassed`: número de casos sin clasificación. Solo se debe usar esta opción en el caso de que la respuesta sea binaria.
- `scaleX`: si es `TRUE`, reescala las variables independientes.
- `pvals.expli`: indica si los p-valores individuales deben ser reportados para la selección del modelo.

Los principales elementos de la salida de esta función son:

- `ww`: pesos antes de la normalización en L_2.
- `wwnorm`: pesos después de la normalización en L_2.
- `tt`: componentes mínimo cuadrado parciales.
- `pp`: coeficiente de cada covariable en cada componente.
- `CoeffC`: coeficiente de cada componente PLS.
- `RSS`: suma de cuadrados residual de la escala original.
- `RSSresidY`: suma de cuadrados residual con las variables reescaladas.
- `R2`: coeficiente R^2 de la escala original.
- `R2residY`: coeficiente R^2 de la escala estandarizada.
- `AIC`: AIC para los modelos con distinto número de componentes.
- `BIC`: BIC para los modelos con distinto número de componentes.
- `Coeffsmodel_vals`: los coeficientes de las componentes para los modelos con distinto número de componentes.
- `CoeffCFull`: matriz de los coeficientes de los predictores.
- `CoeffConstante`: valor del intercepto para los modelos con distinto número de componentes.
- `Std.Coeffs`: vector de coeficientes de la regresión estandarizados.
- `Coeffs`: vector de coeficientes de la regresión con los datos sin reescalar.
- `Yresidus`: residuos del modelo.
- `residusY`: residuos del modelo con los datos estandarizados.

- `InfCrit`: muestra la tabla con los criterios de información (`AIC, BIC, MissClassed, Chi2_Pearson_Y, RSS, R2, R2residY, RSSresidY`).
- `FinalModel`: modelo final con los componentes del PLS.

3.4. Funciones de R para los modelos GAM

En esta sección, explicaremos algunas funciones de R que podemos aplicar para poder llevar a cabo la estimación de modelos GAM en alta dimensión, tanto con variable respuesta continua como binaria.

Para llevar a cabo la regresión del modelo GAM, utilizaremos la función `gam()` de la librería `mgcv` [40]. Esta función utiliza el algoritmo backfitting. Los principales elementos que vamos a utilizar para realizar el ajuste son:

- `formula`: la fórmula especificando las variables que incluímos en el modelo y si están o no suavizadas.
- `family`: especificamos la distribución que vamos a utilizar para el ajuste. Por defecto es la gaussiana.
- `data`: un data frame o una lista que contiene tanto la variable respuesta como el resto de variables.
- `method`: selecciona el método de estimación del parámetro de suavizado. Los distintos métodos que podemos escoger son: `"GCV.Cp"`, utiliza GCV para los parámetros de escala desconocidos, y C_p de Mallow / UBRE / AIC para los parámetros de escala conocidos; `"GACV.Cp"`, es equivalente a `"GCV.Cp"` pero utilizando GACV en lugar de GCV; `"REML"`, para la estimación REML, incluyendo los parámetros de escala desconocidos; `"P-REML"`, para la estimación REML, pero utilizando la estimación del parámetro de escala de Pearson; `"ML"` y `"P-ML"` son similares, pero utilizando máxima verosimilitud en lugar de REML. Para el caso de la familia exponencial, el método `"REML"` es el que utiliza por defecto, siendo la otra única opción el método `"ML"`.

Con esta función obtenemos un objeto de tipo `gam`. Para estudiar los resultados de la estimación, debemos utilizar la función `summary()`, que nos devuelve:

- `p.coeff`: array con las estimaciones de los coeficientes paramétricos del modelo.
- `p.pv`: array con los p-valores para la hipótesis nula de que el parámetro correspondiente es cero.
- `m`: número de términos suavizados en el modelo.
- `chi.sq`: array con los test estadísticos para la significación de los términos suavizados del modelo.
- `s.pv`: array con p-valores aproximados para la hipótesis nula de que cada término suavizado es cero.
- `se`: array de errores estándar estimados para todos los parámetros estimados.

- `r.sq`: el R^2 ajustado del modelo.
- `dev.expl`: la proporción de null deviance explicada por el modelo.
- `edf`: array de grados de libertad estimados para los términos del modelo.
- `residual.df`: grados de libertad estimados.
- `pTerms.df`: los grados de libertad asociados con cada término paramétrico, excluyendo la constante.
- `pTerms.pv`: p-valores para la hipótesis nula de que cada término es cero.
- `p.table`: tabla de significación de los parámetros.
- `s.table`: tabla de significación para los suavizadores.
- `p.Terms`: tabla de significación para los términos paramétricos del modelo.

Pero esta función, tiene la particularidad de que podemos escoger el tipo de suavizado que queremos para cada variable. Para realizar ésto, existen, entre otras, dos funciones que se pueden aplicar a cada variable, o a varias simultaneamente, dentro de la función `gam()`:

- `s()`: aplica una base de splines.
- `te()`: produce un producto tensor suavizado de varias bases de suavizado.

Los elementos de estas funciones son los mismos:

- `bs()`: indicamos la base de suavización que queremos utilizar. `"tp"` para thin plate regression spline, `"ps"` para p-splines, `çr"` para spline cúbico, `.ªd"` para los suavizadores adaptativos, entre otras opciones.
- `by`: una variable numérica o un factor de la misma dimensión que cada variable. Si es numérica, multiplica la suavización evaluada por cada valores de la variable correspondiente; si es un factor, reproduce la suavización para cada nivel del factor.

4. Aplicación a datos reales

En esta sección aplicaremos las funciones explicadas en el capítulo anterior a una base de datos real. En cada apartado se indicará el código utilizado y algunas de las salidas obtenidas, ya que sería incómodo mostrarlas todas por motivos de espacio.

La base de datos que emplearemos contiene información de las características de las viviendas vendidas entre los años 2006 y 2010 en Ames, Iowa [2]. La base de datos contiene un total de 82 variables, de las cuales 23 son nominales, 23 ordinales, 14 discretas, 20 continuas y dos adicionales que sirven como identificadoras. Contiene 2930 viviendas vendidas entre los años de estudio.

Para realizar nuestro análisis se eliminaron las dos primeras variables que representaban la ordenación de las viviendas y el número de identificación de la parcela en la que están construidas. Además, se eliminaron otras nueve variables en la que casi todos los datos eran datos faltantes. Por esta misma razón, se eliminaron también 1771 viviendas. Finalmente, la base de datos que vamos a utilizar tiene unas dimensiones de 72 variables y 1159 viviendas. A pesar de que no cumple la condición de que $p > N$, como el número de covariables es muy elevado, nos va a servir para ilustrar la utilización de las técnicas explicadas anteriormente.

En este apartado aplicaremos los métodos anteriormente explicados para dos casos distintos: en el primero, la variable respuesta va a ser continua; en cambio, en el segundo, la variable respuesta es binaria, es decir, solamente toma los valores 1 (acierto) y 0 (fracaso). La variable respuesta continua de nuestra base de datos es la variable `SalePrice` (precio de la vivienda), que es la que vamos a utilizar también para crear la variable respuesta binaria `Price`. Para crearla, calcularemos el precio medio de todas las viviendas que quedan en la base de datos y a todas aquellas que tengan un valor mayor que la media les asignaremos un 1, y a las que tengan un valor igual o menor que la media le asignaremos un 0.

Utilizar esta variable respuesta binaria artificial en un problema de clasificación no es del todo correcto, porque los grupos deben estar predefinidos y, en nuestro caso, no lo están, ya que son grupos artificiales creados a partir de otra variable. Además, tiene otro problema, y es que son grupos aleatorios, es decir, dependen de la muestra, por lo que si la muestra cambia los grupos también van a ser distintos. Pero, a pesar de estos problemas, lo que se pretende en este capítulo es ilustrar la utilización de distintas técnicas estadísticas sobre una base de datos, por lo que aceptaremos como válida esta variable artificial.

El código en R para realizar las transformaciones de la base de datos y crear la variable respuesta binaria es el siguiente:

```
> predatos <- read.delim("AmesHousing.txt",header=T)
> datos <- predatos[,-c(1,2,8,11,64,65,66,74,75,76)]
> datos <- na.omit(datos)
> mean(datos$SalePrice)
```

```
  [1] 223082.2
> for (i in 1:dim(datos)[1]){
    if(datos$SalePrice[i]<=mean(datos$SalePrice)){
      datos$Price[i] <- 0
    }else{datos$Price[i] <- 1}
  }
> datos <- datos[,-72]
> dim(datos)
  [1] 1159 72
> Covar <- datos[,1:71]
```

Para evitar el bucle, podemos hacer de forma equivalente la siguiente asignación:

```
> datos$Price <- datos$SalePrice > mean(datos$SalePrice)
```

Aunque en este caso nos devuelve `TRUE` o `FALSE`, a efectos de utilizar la variable en las funciones no hay ninguna diferencia. Nuestra matriz X de covariables se va a llamar `Covar`.

4.1. Métodos de selección de subconjuntos

En este apartado, aplicaremos los distintos métodos de selección de subconjuntos explicados anteriormente a nuestra base de datos. Utilizaremos para ello la función `regsubsets()` explicada en el capítulo anterior. Esta función solamente realiza los dos primeros pasos de los tres métodos (best subset selection, forward-stepwise y backward-stepwise). Es decir, ajusta todos los modelos posibles para cada número de covariables y encuetra el que tiene una menor RSS o mayor R^2 para cada número de covariables. Por eso, después de aplicar la función, debemos seleccionar el mejor modelo de entre todos los que nos da la función. Para la selección del modelo para cada número de covariables, esta función puede utilizar cuatro criterios: RSS, C_p de Mallows, BIC y R^2 ajustado.

En el caso del método forward con respuesta continua, la llamada que hacemos a la función `regsubsets` es:

```
> library(leaps)
> bsscf <- regsubsets(SalePrice~.,data=datos,nvmax=500,method="forward",really.big=TRUE)
> bsscf.sum=summary(bsscf)
```

Le indicamos que el número máximo de variables a incluir en el modelo sean 500, y le indicamos `really.big=TRUE`. Para obtener el mejor modelo de entre todos los que nos da esta función con los distintos criterios, utilizamos los siguientes comandos:

```
> which.min(bsscf.sum$rss)

[1] 501

> which.min(bsscf.sum$cp)

[1] 118

> which.min(bsscf.sum$bic)

[1] 61

> which.max(bsscf.sum$adjr2)

[1] 215
```

Vemos que los mejores modelos con los cuatro criterios son muy distintos. Es muy elevado el número de variables resultantes en el modelo, exceptuando con el BIC que es de 61, que es alto pero no en comparación con los otros tres métodos. En este caso, los coeficientes resultantes son:

```
> coef(bsscf,61)

        (Intercept)      MS.SubClass120    MS.ZoningC (all)           Lot.Area
       2.442408e+05       -3.185563e+04       -4.817132e+04       1.324393e+00
       Lot.ShapeIR2     Land.ContourLvl       Lot.ConfigFR2  NeighborhoodBrDale
       3.110592e+04       -2.515790e+03       -1.183621e+04       -7.186732e+04
 NeighborhoodGreens  NeighborhoodNWAmes NeighborhoodOldTown NeighborhoodSawyerW
       4.320873e+04       -2.626634e+04       -2.484949e+04       3.089141e+03
NeighborhoodSomerst NeighborhoodStoneBr NeighborhoodVeenker    Condition.1Feedr
       2.458750e+04        9.578169e+04        1.140141e+04       -3.298886e+04
    Condition.1Norm     Condition.1RRAe   House.Style2.5Fin       Overall.Qual7
       9.799712e+03       -5.520764e+04       -1.587377e+04       -1.557185e+04
      Overall.Cond8       Overall.Cond9      Year.Built1880      Year.Built1882
       1.033718e+04        3.283009e+04        1.265954e+05       -3.641775e+04
     Year.Built1890      Year.Built1892      Year.Built1893      Year.Built1900
       3.944182e+04        2.870574e+05        1.542031e+05       -3.593268e+04
     Year.Built1904  Year.Remod.Add1959    Roof.MatlWdShake   Exterior.1stStone
       2.112007e+04       -5.483210e+04        1.977930e+04       -1.406494e+04
 Exterior.1stStucco Exterior.1stWdShing Exterior.2ndAsphShn Exterior.2ndBrk Cmn
      -2.003029e+04       -2.949058e+04       -1.880714e+04       -3.957409e+04
Exterior.2ndHdBoard   BsmtFin.Type.1BLQ   BsmtFin.Type.1Rec        BsmtFin.SF.1
      -2.293496e+04       -3.795610e+04       -4.156612e+04        5.726890e+01
  BsmtFin.Type.2Rec         HeatingGasA     Low.Qual.Fin.SF     Bsmt.Full.Bath1
      -2.583271e+04       -3.387658e+04        1.235877e+01        7.460694e+02
     Bedroom.AbvGr1      TotRms.AbvGrd4      Fireplace.QuGd   Garage.TypeDetchd
      -5.716275e+03       -4.813662e+04        1.767115e+04       -5.368281e+04
  Garage.Yr.Blt1910   Garage.Yr.Blt1920   Garage.Yr.Blt1938   Garage.Yr.Blt1939
```

```
     -1.174079e+05       -2.643493e+03       -7.235117e+04       -2.271340e+04
  Garage.Yr.Blt1949   Garage.Yr.Blt1950        Sale.TypeWD   Year.Remod.Add1953
     -4.289662e+04       -1.793722e+04       -3.646498e+04       -5.029692e+04
    Roof.MatlMetal Exterior.2ndPreCast         Bsmt.CondPo         Bsmt.CondTA
      0.000000e+00        0.000000e+00       -6.314447e+04        9.565264e+03
  Garage.Yr.Blt1957        Sale.TypeOth
     -4.235869e+04        0.000000e+00
```

En el caso del método backward con respuesta continua, la llamada a la función es la siguiente:

```
> bsscb <- regsubsets(SalePrice~., data=datos, nvmax=500, method="backward", really.big=TRUE)
> bsscb.sum=summary(bsscb)
> which.min(bsscb.sum$rss)

[1] 501

> which.min(bsscb.sum$cp)

[1] 121

> which.min(bsscb.sum$bic)

[1] 57

> which.max(bsscb.sum$adjr2)

[1] 268
```

En este caso, también obtenemos modelos con un número muy elevado de variables, siendo, como en el método anterior, el criterio BIC con el que obtenemos el menor número con 57. Los coeficientes de las covariables escogidas en este caso son los siguientes:

```
> coef(bsscb,57)

        (Intercept)     MS.ZoningC (all)          StreetPave         Lot.ShapeIR2
       268798.87554         -46902.92007          47696.49829          37052.04266
    Land.ContourLvl        Lot.ConfigFR2        Lot.ConfigFR3   NeighborhoodBrDale
         1740.22699          -7045.93070         -24155.62204         -77961.98486
  NeighborhoodGreens  NeighborhoodNPkVill   NeighborhoodNWAmes  NeighborhoodOldTown
       -30436.28724           -198.95849         -11570.87410         -56523.30109
 NeighborhoodSomerst  NeighborhoodStoneBr     Condition.1Feedr      Condition.1Norm
        16225.70946          77759.04176         -20542.79406           9167.73041
    Condition.1RRAn    House.Style2.5Fin        Overall.Cond9       Year.Built1880
          451.13192          13256.90206          -7372.93630         120646.25948
     Year.Built1882       Year.Built1890       Year.Built1892       Year.Built1893
```

```
        37546.61482          38643.51097         198421.61134         182150.40713
     Year.Built1900    Year.Remod.Add1951    Year.Remod.Add1955    Year.Remod.Add1958
        24879.78578          -36954.28727         -75831.76706         -39531.55542
    Year.Remod.Add1959      Roof.MatlWdShake  Exterior.1stBrkFace     Exterior.1stStone
        -37978.44066          32684.85804          11974.76157         -83433.25982
   Exterior.1stStucco  Exterior.1stWdShing  Exterior.2ndAsphShn  Exterior.2ndBrk Cmn
        -33047.92339         -20002.53532           8848.69213         -46844.72962
  Exterior.2ndHdBoard     BsmtFin.Type.1BLQ         BsmtFin.SF.1          HeatingGasA
         -9284.59806         -22685.28602             47.63462         -42449.30296
      Bsmt.Full.Bath1       Bedroom.AbvGr1       Kitchen.QualGd       Kitchen.QualTA
         -4561.66294         -30729.84678         -83419.85107        -139902.92458
         Fireplaces3          Fireplaces4    Garage.Yr.Blt1939    Garage.Yr.Blt1949
         -3450.69570          21660.64638         -32842.40340         -26370.91981
Sale.ConditionAlloca       Condition.2RRNn       Bldg.TypeDuplex     House.Style1.5Unf
         77121.66340              0.00000         -82825.15565         -47406.73034
       Roof.MatlMetal  Exterior.2ndPreCast          Bsmt.CondPo          Bsmt.CondTA
             0.00000              0.00000         36288.69325          14379.38346
    Garage.Yr.Blt1957         Sale.TypeOth
        -16968.29267              0.00000
```

En el caso de best subset selection, la llamada a la función es la siguiente:

```
> bssc <- regsubsets(SalePrice~., data=datos, method="exhaustive", really.big=TRUE)
```

Pero no es eficiente utilizar este método ya que, fijando el número máximo de variables para incorporar al modelo en 2, el tiempo de computación es de cinco minutos, pero con solamente 3 variables, el tiempo de computación es de varias horas, por lo que es impensable utilizar este método para encontrar el mejor modelo de entre más de 70 variables como en los casos anteriores.

En la Tabla 1, podemos ver una comparación de las variables que obtenemos con ambos métodos, con el criterio de selección BIC. Vemos que, tanto en el caso de Forward- como de Backwars-stepwise, el número de variables escogidas para el modelo óptimo es muy elevado. Vemos que los dos modelos obtenidos comparten 20 variables, de las 30 y 26 que tienen los modelos Forward y Backward, respectivamente.

Variable	Forward	Backward	Descripción
`Intercept`	Si	No	
`MS.SubClass`	Si	No	Tipo de vivienda.
`MS.Zoning`	Si	Si	Identifica el tipo de zona de la vivienda.
`Street`	No	Si	Tipo de carretera de acceso a la vivienda.
`Lot.Area`	Si	No	Tamaño del terreno de la vivienda.

Variable	Forward	Backward	Descripción
`Lot.Shape`	Si	Si	Forma de la propiedad.
`Land.Contour`	Si	Si	Llanura del terreno.
`Lot.Config`	Si	No	Posición del terreno.
`Neighborhood`	Si	Si	Barrio.
`Condition1`	Si	Si	Proximidad de la vivienda a distintos servicios.
`Condition.2`	No	Si	Proximidad de la vivienda a distintos servicios, si hay más de uno.
`Bldg.Type`	No	Si	Tipo de vivienda en el sentido de familiar, duplex, etc.
`House.Style`	Si	Si	Estilo de la vivienda.
`Overall.Qual`	Si	No	Calificación de los materiales y el nivel de terminado de la vivienda.
`Overall.Cond`	Si	Si	Calificación de la vivienda en general.
`Year.Built`	Si	Si	Año de contrucción.
`Year.Remod.Add`	Si	Si	Año de restauración.
`Roof.Matl`	Si	Si	Tipo de tejado.
`Exterior.1st`	Si	Si	Material del tejado.
`Exterior.2nd`	Si	Si	Material del tejado, en caso de haber más de uno.
`Bmst.Cond`	Si	Si	Condición general del sótano.
`BsmtFin.Type.1`	Si	Si	Calificación del nivel de terminado del sótano.
`BsmtFin.SF.1`	Si	Si	Pies cuadrados de `BsmtFin.Type.1`.
`BsmtFin.Type.2`	Si	No	Calificación del nivel de terminado del sótano, si hay más de una.
`HeatingGas`	Si	Si	Tipo de calefacción.
`Low.Qual.Fin.SF`	Si	No	Pies cuadrados de materiales de baja calidad en todas las plantas.
`Bsmt.Full.Bath`	Si	Si	Número de baños completos en el sótano.
`Bedroom.AbvGr`	Si	Si	Dormitorios por encima del nivel del suelo, sin incluir las del sótano.
`Kitchen.Qual`	No	Si	Calidad de la cocina.
`TotRms.AbvGrd`	Si	No	Habitaciones por encima del nivel del suelo, sin contar los baños.
`Fireplace.Qu`	Si	No	Calidad de las chimeneas.
`Fireplaces`	No	Si	Número de chimeneas.
`Garage.TypeDet`	Si	No	Localización del garage con respecto al resto de la vivienda.
`Garage.Yr.Blt`	Si	Si	Año de construcción del garage.

Variable	Forward	Backward	Descripción
`Sale.Type`	Si	Si	Tipo de venta.
`Sale.Condition`	No	Si	Condiciones de la venta.

Tabla 1: Variables incluídas en los modelos Forward- y Backward-stepwise.

4.2. Métodos de regularización

En este apartado aplicaremos a nuestra base de datos las funciones explicadas en el capítulo anterior para la regresión ridge, lasso y Least Angle. Aplicaremos estas funciones tanto al caso de regresión, como al de clasificación. En primer lugar, comenzaremos con la regresiones ridge y lasso.

4.2.1. Regresión Ridge y lasso

Como ya comentamos, la función que utilizaremos para realizar las regresiones ridge y lasso es `gmlnet()`. Utilizaremos la función `model.matrix()` para construir la matriz de diseño.

```
> Resto <- model.matrix(Price~.,datos)[,-1]
> dim(Resto)
```

```
[1] 1159  567
```

Al tener variables dicotómicas y ordinales, la función las sustituye por el número de variables dummy correspondientes. Por eso, nuestra matriz de diseño `Resto` va a tener 567 columnas, en lugar de las 71 que tenía la matriz `Covar`.

4.2.1.1. Variable respuesta continua

Para el caso de la regresión ridge con variable respuesta continua, en primer lugar, estimaremos la función de regresión y, a continuación, utilizaremos la función `cv.glmnet()` para encontrar el mejor λ mediante validación cruzada. Encontraremos dos λ: el primero (`lambda.rrc`), será el lambda que minimice la validación cruzada media, y el segundo (`lambda.rrc.1se`), será el mayor λ tal que su error se encuentra a menos de un error estándar del que obtenemos con el λ mínimo (`lambda.rrc`). Después, extraeremos los coeficientes de cada modelo con la función `predict()`.

```
> library(glmnet)
> rrc <- glmnet(Resto,SalePrice,alpha=0)
```

```
> cv.rrc <- cv.glmnet(Resto,SalePrice,alpha=0)
> lambda.rrc <- cv.rrc$lambda.min; lambda.rrc

[1] 93185.89

> lambda.rrc.1se <- cv.rrc$lambda.1se; lambda.rrc.1se

[1] 342773.1

> coef.rrc <- predict(rrc,type="coefficients",s=lambda.rrc)
> coef.rrc

568 x 1 sparse Matrix of class "dgCMatrix"
                                 1
(Intercept)           1.299043e+05
MS.SubClass30        -7.281068e+03
MS.SubClass40        -6.155896e+03
MS.SubClass45         3.045640e+03
MS.SubClass50        -1.684593e+03
MS.SubClass60         1.064527e+03
MS.SubClass70         2.216700e+03
MS.SubClass75         6.697196e+03
MS.SubClass80        -2.516018e+03
MS.SubClass85        -2.951872e+03
MS.SubClass90        -6.404904e+03
MS.SubClass120       -6.019244e+03
MS.SubClass160       -1.769597e+03
MS.SubClass180       -1.489161e+04
MS.SubClass190       -6.496435e+03
MS.ZoningC (all)     -2.284012e+04
MS.ZoningFV           5.910576e+03
MS.ZoningI (all)      .
MS.ZoningRH          -2.326897e+03
MS.ZoningRL           1.659553e+03
MS.ZoningRM          -4.887520e+03
Lot.Frontage          5.590625e+01
Lot.Area              3.391302e-01
StreetPave            1.102290e+04

> coef.rrc.1se <- predict(rrc,type="coefficients",s=lambda.rrc.1se)
> coef.rrc.1se

568 x 1 sparse Matrix of class "dgCMatrix"
                                 1
(Intercept)           1.673093e+05
MS.SubClass30        -6.259824e+03
MS.SubClass40        -4.990682e+03
MS.SubClass45        -1.204915e+03
MS.SubClass50        -2.070813e+03
```

```
MS.SubClass60          1.865353e+03
MS.SubClass70          7.530631e+02
MS.SubClass75          4.906570e+03
MS.SubClass80         -2.570025e+03
MS.SubClass85         -3.161808e+03
MS.SubClass90         -4.255036e+03
MS.SubClass120        -2.778104e+03
MS.SubClass160        -2.179351e+03
MS.SubClass180        -1.067487e+04
MS.SubClass190        -4.201571e+03
MS.ZoningC (all)      -1.007062e+04
MS.ZoningFV            2.761252e+03
MS.ZoningI (all)       .
MS.ZoningRH           -4.197383e+03
MS.ZoningRL            1.656057e+03
MS.ZoningRM           -3.381939e+03
Lot.Frontage           6.461984e+01
Lot.Area               2.393582e-01
StreetPave             4.538318e+03
```

En ambos casos solamente mostramos 24 de las 567 variables más el intercepto ya que, como sabemos, la regresión ridge no elimina variables del modelo sino que solamente reduce sus coeficientes hacia cero. Vemos que, por ejemplo, en el caso de la variable `Street` (tipo de carretera de acceso a la vivienda), escogiendo el λ mínimo (`lambda.rrc`), que la carretera esté pavimentada (`StreetPave`), frente a la opción de que esté en gravilla (categoría de referencia), incrementa el precio de la vivienda en aproximadamente 11.022$. En cambio, si escogemos el `lambda.rrc.1se`, que la carretera de entrada esté pavimentada solamente incrementa el precio de la vivienda en 4.538$. A pesar de esta diferencia, podemos ver que en ambos casos tiene un efecto positivo sobre el precio.

En este ejemplo podemos apreciar lo que explicábamos anteriormente. El `lambda.rrc` es menor que el `lambda.rrc.1se`, por lo que los coeficientes se reducen en mayor medida en el segundo caso.

Para el caso del método lasso con respuesta continua, la llamada a la función es la siguiente:

```
> lassoc <- glmnet(Resto,SalePrice)
```

Vemos que en este caso, no le especificamos el `alpha` ya que por defecto es 1, que se corresponde con el método lasso.

```
> cv.lassoc <- cv.glmnet(Resto,SalePrice)
> lambda.lassoc <- cv.lassoc$lambda.min; lambda.lassoc
```

```
[1] 1518.698

> lambda.lassoc.1se <- cv.lassoc$lambda.1se; lambda.lassoc.1se

[1] 10714.11

> coef.lassoc <- predict(lassoc,type="coefficients",s=lambda.lassoc)
> coef.lassoc

568 x 1 sparse Matrix of class "dgCMatrix"
                                    1
(Intercept)             6.201887e+04
MS.SubClass120         -2.924231e+03
MS.SubClass190         -3.955243e+03
MS.ZoningRM            -5.744688e+03
Lot.Area                3.893959e-01
Lot.ShapeIR2            1.345624e+03
Lot.ShapeIR3           -7.320064e+03
Land.ContourHLS         6.675763e+03
Lot.ConfigCulDSac       1.055790e+04
Lot.ConfigFR3          -2.487758e+03
Land.SlopeMod           1.037110e+03
NeighborhoodCrawfor     9.573696e+03
NeighborhoodEdwards    -2.090028e+04
NeighborhoodMeadowV    -5.858197e+03
NeighborhoodNAmes      -5.498960e+03
NeighborhoodNoRidge     1.855823e+04
NeighborhoodNridgHt     1.743981e+04
NeighborhoodOldTown    -9.039069e+03
NeighborhoodSomerst     9.124297e+03
NeighborhoodStoneBr     3.114575e+04
NeighborhoodSWISU      -1.074623e+04
Condition.1Feedr       -8.416700e+03
Condition.1Norm         4.821401e+03
Condition.1PosA         4.682602e+03
Condition.2PosA         1.817049e+04
Condition.2PosN        -6.344781e+04
Bldg.Type2fmCon        -2.328911e+01
Bldg.TypeTwnhs         -2.930062e+02
Bldg.TypeTwnhsE        -1.782570e+03
Overall.Qual4          -3.397433e+03
Overall.Qual5          -5.563200e+03
Overall.Qual8           1.159077e+04
Overall.Qual9           5.395624e+04
Overall.Qual10          7.450575e+04
Overall.Cond3          -1.975038e+04
Overall.Cond4          -1.203974e+04
Overall.Cond7           8.789033e+02
Overall.Cond8           3.396674e+03
```

```
Year.Built1910        -1.145015e+04
Year.Built1935         1.898143e+04
Year.Built1992        -2.911051e+02
Year.Remod.Add1970    -1.574906e+03
Year.Remod.Add1984     5.786659e+03
Year.Remod.Add1989    -2.733802e+03
Year.Remod.Add1996     4.064040e+03
Year.Remod.Add2007     5.388842e+03
Year.Remod.Add2010     5.408049e+03
Roof.MatlCompShg       1.116660e+04
Roof.MatlWdShngl       4.942731e+04
Exterior.1stBrkFace    9.512512e+03
Exterior.1stCemntBd    8.655872e+02
Exterior.1stStucco    -8.268320e+03
Exterior.2ndImStucc    1.492607e+04
Mas.Vnr.Area           1.839735e+01
Exter.QualTA          -1.476561e+04
Exter.CondFa          -4.024893e+03
FoundationPConc        4.873434e+03
FoundationStone        4.420290e+03
Bsmt.QualEx            1.849324e+04
Bsmt.ExposureGd        1.238855e+04
Bsmt.ExposureNo       -3.685708e+03
BsmtFin.Type.1GLQ      8.403344e+03
BsmtFin.Type.1Rec     -7.658455e+02
BsmtFin.SF.1           8.528283e+00
BsmtFin.SF.2           2.744761e+00
Total.Bsmt.SF          1.506621e+00
Heating.QCFa          -5.217885e+03
Heating.QCTA          -2.365170e+03
Central.AirY           4.767671e+03
X1st.Flr.SF            1.729080e+01
Gr.Liv.Area            4.303493e+01
Bsmt.Full.Bath1        5.135144e+03
Full.Bath1            -5.182508e+03
Full.Bath3             2.309544e+04
Half.Bath1             5.750055e+03
Half.Bath2            -4.221562e+03
Bedroom.AbvGr3        -9.865501e+02
Bedroom.AbvGr4         2.683062e+03
Kitchen.QualFa        -2.032716e+02
Kitchen.QualTA        -3.984671e+03
TotRms.AbvGrd10        1.855292e+03
TotRms.AbvGrd12       -1.747726e+04
TotRms.AbvGrd15       -3.947490e+05
FunctionalTyp          6.228392e+03
Fireplaces2            1.288427e+04
Fireplaces3           -3.090366e+04
```

```
Garage.Yr.Blt1992     -5.722549e+03
Garage.Yr.Blt2003      1.725799e+03
Garage.Yr.Blt2009      2.706324e+04
Garage.FinishFin       9.490124e+02
Garage.FinishUnf      -3.899679e+02
Garage.Cars3           2.582329e+04
Wood.Deck.SF           1.401945e+00
X3Ssn.Porch            1.010465e+00
Screen.Porch           4.965350e+01
Sale.TypeCon           6.253409e+03
Sale.TypeNew           4.303230e+03
> coef.lassoc.1se <- predict(lassoc,type="coefficients",s=lambda.lassoc.1se)
> coef.lassoc.1se
568 x 1 sparse Matrix of class "dgCMatrix"
                                1
(Intercept)           103365.126929
Overall.Qual9          24274.675135
Overall.Qual10         13066.442938
Mas.Vnr.Area              16.176221
Exter.QualTA          -23279.277886
FoundationPConc         2499.762970
Bsmt.QualEx            36310.941358
Bsmt.ExposureGd         8239.876715
BsmtFin.Type.1GLQ      11484.651362
Total.Bsmt.SF             10.898614
X1st.Flr.SF                5.992589
Gr.Liv.Area               49.020256
Kitchen.QualTA         -3345.081102
TotRms.AbvGrd15       -17089.370514
Garage.Cars3           34937.105368
```

En ambos grupos de coeficientes se muestran todas las variables cuyos coeficientes son distintas de cero. Podemos observar que, como ya sabíamos por la forma de las penalizaciones de la regresión ridge y lasso, se da en ambos métodos la misma situación: como el `lambda.lassoc` es menor que el `lambda.lassoc.1se`, la penalización es menor en el primer caso y, por lo tanto, el número de covariables que son iguales a cero también es menor en el primer caso. En el caso de la estimación lasso con el `lambda.lassoc`, el número de variables que permanecen en el modelo son 97, mientras que, con el `lambda.lassoc.1se`, son solamente 15, en ambos casos incluyendo el intercepto.

4.2.1.2. Variable respuesta binaria

En el caso de variable respuesta binaria, utilizaremos la misma función que para el caso de respuesta continua, y realizaremos el mismo proceso para encontrar los mejores λ. La llamada a la función `glmnet()` que hemos realizado para realizar la regresión ridge es:

```
> rr <- glmnet(Resto, Price, family="binomial", alpha=0)
> cv.rr <- cv.glmnet(Resto, Price, family="binomial", alpha=0)
> lambda.rr <- cv.rr$lambda.min; lambda.rr

[1] 0.05690311

> lambda.rr.1se <- cv.rr$lambda.1se; lambda.rr.1se

[1] 0.2297189

> coef.rr <- predict(rr, type="coefficients", s=lambda.rr)
> coef.rr

568 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)          -5.973799
MS.SubClass30        -0.174177
MS.SubClass40        -0.070790
MS.SubClass45        -0.047465
MS.SubClass50        -0.044461
MS.SubClass60        -0.004831
MS.SubClass70         0.249005
MS.SubClass75         0.020133
MS.SubClass80        -0.251459
MS.SubClass85        -0.217311
MS.SubClass90        -0.132928
MS.SubClass120       -0.370495
MS.SubClass160        0.104848
MS.SubClass180       -0.211410
MS.SubClass190       -0.593558
MS.ZoningC (all)     -0.780838
MS.ZoningFV           0.511507
MS.ZoningI (all)      .
MS.ZoningRH          -0.562792
MS.ZoningRL          -0.030015
MS.ZoningRM          -0.166957
Lot.Frontage          0.004620
Lot.Area              0.000008
StreetPave            0.923342

> coef.rr.1se <- predict(rr, type="coefficients", s=lambda.rr.1se)
> coef.rr.1se

568 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)          -3.383584
MS.SubClass30        -0.152276
MS.SubClass40        -0.095000
MS.SubClass45        -0.069212
```

```
MS.SubClass50         -0.053468
MS.SubClass60          0.036862
MS.SubClass70          0.127086
MS.SubClass75          0.081967
MS.SubClass80         -0.194548
MS.SubClass85         -0.163939
MS.SubClass90         -0.154235
MS.SubClass120        -0.195351
MS.SubClass160        -0.008412
MS.SubClass180        -0.152073
MS.SubClass190        -0.302790
MS.ZoningC (all)      -0.421202
MS.ZoningFV            0.321735
MS.ZoningI (all)       .
MS.ZoningRH           -0.358492
MS.ZoningRL           -0.021196
MS.ZoningRM           -0.102121
Lot.Frontage           0.003374
Lot.Area               0.000007
StreetPave             0.460985
```

Igual que ocurría en el caso de variable respuesta continua, al ser mayor el `labmda.rr.1se` que el `lambda.rr`, la penalización que se produce en el primer caso es mayor que en el segundo. Solamente presentamos los primeros 24 coeficientes como muestra ya que, como se indica en las salidas, en ambos casos hay 568 coeficientes (28 de ellos nulos), por lo que la interpretación de los mismos es muy compleja.

En el caso del método lasso, los comandos utilizados para realizar la estimación son los siguientes:

```
> lasso <- glmnet(Resto, Price, family="binomial")
> cv.lasso <- cv.glmnet(Resto, Price, family="binomial")
> lambda.lasso <- cv.lasso$lambda.min; lambda.lasso

[1] 0.03491522

> lambda.lasso.1se <- cv.lasso$lambda.1se; lambda.lasso.1se

[1] 0.04615587

> coef.lasso <- predict(lasso, type="coefficients", s=lambda.lasso)
> coef.lasso

568 x 1 sparse Matrix of class "dgCMatrix"
                              1
(Intercept)          -4.533825
```

```
Lot.Area                0.000003
NeighborhoodCrawfor     0.071253
NeighborhoodSomerst     0.148381
Overall.Qual6          -0.030519
Overall.Qual8           0.942438
Overall.Qual9           0.690316
Mas.Vnr.Area            0.000095
Exter.QualTA           -0.648468
Bsmt.QualEx             0.444089
Bsmt.QualTA            -0.031463
BsmtFin.Type.1GLQ       0.332443
Total.Bsmt.SF           0.000423
X1st.Flr.SF             0.000178
Gr.Liv.Area             0.001754
Bsmt.Full.Bath1         0.120740
Kitchen.QualTA         -0.700977
Garage.Cars3            0.875940
```

```
> coef.lasso.1se <- predict(lasso, type="coefficients", s=lambda.lasso.1se)
> coef.lasso.1se
```

```
568 x 1 sparse Matrix of class "dgCMatrix"
                               1
(Intercept)           -3.923134
Overall.Qual8          0.805938
Overall.Qual9          0.424608
Exter.QualTA          -0.682553
Bsmt.QualEx            0.439626
BsmtFin.Type.1GLQ      0.313039
Total.Bsmt.SF          0.000394
X1st.Flr.SF            0.000172
Gr.Liv.Area            0.001520
Kitchen.QualTA        -0.602771
Garage.Cars3           0.853291
```

En ambos casos solamente presentamos las covariables que se mantienen en el modelo. Vemos que en el caso del lasso, al incorporar selección de variables, con ambos λ, el número de covariables que quedan en el modelo es muy reducido. Además, igual que pasaba con la regresión ridge, al ser el `lambda.lasso` menor que el `lambda.lasso.1se`, la reducción de las variables también es menor y, por lo tanto, el número de covariables que deja en el modelo es mayor. Para el caso del `lambda.lasso` encontrado por el método de validación cruzada, nos quedamos solamente con 17 variables más el intercepto, mientras que para el `lambda.lasso.1se` nos quedamos con 12 más el intercepto. Cada una de las variables y su significado se presentan en la Tabla 2.

Variable	`lambda.lasso`	`lambda.lasso.1se`	Descripción
(Intercept)	Si	Si	
Lot.Area	Si	No	Metros cuadrados de la propiedad
NeighborhoodCrawfor	Si	No	La propiedad se encuentra en el barrio Crawford
NeighborhoodSomerst	Si	Si	La propiedad se encuentra en el barrio Somerset
Overall.Qual6	Si	No	Ratio de los materiales y el estado de la vivienda: por encima de la media
Overall.Qual8	Si	Si	Ratio de los materiales y el estado de la vivienda: muy bueno
Overall.Qual9	Si	Si	Ratio de los materiales y el estado de la vivienda: excelente
Mas.Vnr.Area	Si	No	(Continua) Pies cuadrados de fachazada terminada con materiales de alta calidad
Exter.QualTA	Si	Si	(Ordinal) Evalúa la calidad de los materiales del exterior: en la media.
Bsmt.QualEx	Si	Si	(Ordinal) Evalúa la calidad de los materiales del exterior: excelente
Bsmt.QualTA	Si	No	(Ordinal) Evalúa la altura del sótano: en la media (2-2.26 metros)
BsmtFin.Type.1GLQ	Si	Si	(Ordinal) Evalúa el nivel de acabado del sótano: bueno para vivir
Total.Bsmt.SF	Si	Si	(Continua) Pies cuadrados del sótano
X1st.Flr.SF	Si	Si	(Continua) Pies cuadrados del primer piso
Gr.Liv.Area	Si	Si	(Continua) Pies cuadrados del salón
Bsmt.Full.Bath1	Si	Si	(Discreta) Número de baños completos en el sótano: 1
Kitchen.QualTA	Si	Si	(Ordinal) Calidad de la cocina: en la media
Garage.Cars3	Si	Si	(Discreta) Capacidad del garaje: 3 coches

Tabla 2: Variables incluídas en el modelo lasso para los dos `lambda` utilizados.

4.2.2. Least Angle Regression

En este apartado aplicaremos el método Least Angle Regression a nuestra base de datos.

4.2.2.1. Variable respuesta continua

Como hemos comentado anteriormente, utilizaremos la función `lars()` para el caso de variable respuesta continua. A continuación, encontraremos los modelos que minimizan la suma de residuos al cuadrado y el C_p de Mallow. Los comandos utilizados para realizar este proceso son:

```
> library(lars)
> larc <- lars(Resto,SalePrice,type="lar")
> min(larc$RSS); which.min(larc$RSS)
```

```
[1] 474329765543
514

> min(larc$Cp); which.min(larc$Cp)

[1] 315.8383
286
```

Vemos que el modelo que tiene la menor RSS es el que tiene 514 variables, mientras que el que tiene un menor valor del criterio C_p de Mallow es el de 286 variables, en ambos casos sin contar el intercepto. Vemos que resultan modelos con un número muy alto de covariables, tanto que incluso el de menor RSS solamente eliminó 53 de las 567 variables del modelo.

4.2.2.2. Variable respuesta binaria

En este caso, utilizaremos la función `dglars()` explicada en el capítulo anterior. En este caso, solamente utilizaremos el criterio AIC y los grados de libertad `gdf` y `df` para seleccionar el mejor modelo mediante la función `summary()`.

```
> library(dglars)
> lar <- dglars(Price~., data=datos, family="binomial")
> summary(lar, complexity="gdf", K="AIC")

Coefficients:

               Int.        MS.SubClass120          MS.ZoningFV
         -1.423e+00            -6.465e-01            3.861e-01
           Lot.Area       Land.ContourLow     Lot.ConfigCulDSac
          9.268e-06             2.723e-01            4.949e-01
      Lot.ConfigFR3         Land.SlopeMod   NeighborhoodClearCr
         -4.953e-01             3.070e-02            9.520e-01
NeighborhoodCrawfor   NeighborhoodGilbert    NeighborhoodGreens
          9.293e-01            -3.442e-01           -1.098e+00
NeighborhoodMitchel   NeighborhoodNoRidge   NeighborhoodSomerst
          2.766e-01             7.526e-01            2.789e-02
  NeighborhoodSWISU      Condition.1Feedr       Condition.1PosA
         -3.939e-01            -5.028e-01            8.453e-02
    Condition.2PosN       Bldg.Type2fmCon       House.StyleSLvl
         -7.807e-01            -1.395e-02           -6.234e-02
      Overall.Qual5         Overall.Qual6         Overall.Qual8
         -1.396e+00            -5.497e-01            1.524e+00
      Overall.Qual9         Overall.Cond7        Year.Built1880
          3.752e+00             1.343e-01            1.197e+00
     Year.Built1935        Year.Built1954        Year.Built1970
          9.482e-01             5.278e-01            7.738e-02
```

```
     Year.Built1984        Year.Built1991        Year.Built2009
          6.989e-01             1.579e+00             1.450e+00
 Year.Remod.Add1984    Year.Remod.Add1987    Year.Remod.Add1988
          3.043e-01             2.338e-01             1.160e-01
 Year.Remod.Add1989    Year.Remod.Add1997     Roof.StyleMansard
         -5.705e-01             8.682e-01             3.761e-01
Exterior.1stBrkFace   Exterior.1stCemntBd   Exterior.1stHdBoard
          6.572e-01             2.236e-01            -3.407e-01
  Exterior.1stStone   Exterior.2ndWd Shng         Mas.Vnr.Area
          1.946e+00            -4.658e-02             1.175e-03
       Exter.QualTA     BsmtFin.Type.1GLQ          BsmtFin.SF.1
         -4.902e-01             2.057e-01             4.693e-04
       Heating.QCTA       Bsmt.Full.Bath1            Full.Bath1
         -2.810e-01             3.095e-01            -9.668e-01
     Bedroom.AbvGr2        Bedroom.AbvGr4        Bedroom.AbvGr5
         -2.861e-02             2.811e-01             3.969e-01
     Kitchen.QualTA        TotRms.AbvGrd4        TotRms.AbvGrd6
         -9.890e-01            -1.313e-01            -2.249e-01
     TotRms.AbvGrd8        TotRms.AbvGrd9       TotRms.AbvGrd10
          2.166e-01             1.117e+00             6.475e-01
    TotRms.AbvGrd15           Fireplaces2        Fireplace.QuFa
         -4.955e+00             5.419e-01            -2.655e-01
  Garage.Yr.Blt1923     Garage.Yr.Blt1932     Garage.Yr.Blt1954
          6.623e-01             6.045e-01             8.455e-02
  Garage.Yr.Blt1963     Garage.Yr.Blt1977     Garage.Yr.Blt1987
         -7.950e-01            -3.990e-01             1.442e-02
  Garage.Yr.Blt1992      Garage.FinishUnf          Garage.Cars3
         -4.820e-01            -9.513e-03             1.344e+00
      Open.Porch.SF          Screen.Porch             Mo.Sold10
          1.221e-03             2.599e-03            -8.887e-02
       Sale.TypeCWD  Sale.ConditionFamily
         -3.610e-02            -2.410e-01

> summary(lar, complexity="df", K="AIC")

Coefficients:

           Int.  NeighborhoodCrawfor  NeighborhoodNoRidge
     -0.6231135             0.1553894            0.1188168
  Overall.Qual5        Overall.Qual6        Overall.Qual8
     -0.8006088            -0.3929130            0.9765626
  Overall.Qual9         Mas.Vnr.Area         Exter.QualTA
      1.5403903             0.0005300           -0.3998444
    BsmtFin.SF.1           Full.Bath1       Bedroom.AbvGr4
      0.0002541            -0.5680388            0.0703801
 Kitchen.QualTA       TotRms.AbvGrd6       TotRms.AbvGrd9
     -0.8093005            -0.0555467            0.4588128
TotRms.AbvGrd15          Fireplaces2         Garage.Cars3
     -0.2567484             0.1812317            1.3676287
```

```
Screen.Porch
   0.0003621
```

Podemos ver que entre los métodos de complejidad `gdf` y `df` hay una gran diferencia en el número de variables que incluimos en el modelo. En el primer caso, nos quedamos con 77 variables, mientras que en el segundo nos quedamos con solamente 19. Al cambiar del método `BIC` al `AIC` no se aprecia ningún cambio en los resultados.

En conclusión, pudimos ver que los tres métodos se pueden aplicar para realizar regresión con un gran número de covariables. Pero es el método lasso con el que optenemos unos mejores resultados ya que, la regresión ridge, no llega a eliminar variables del modelo sino que reduce sus coeficientes, lo que hace que el modelo resultante sea muy complejo de interpretar. Con la regresión LAR sucede algo parecido, pero en este caso depende del método que fijemos, ya que cuando utilizamos los grados de libertad generalizados (`"gdf"`) en `complexity`, sucede lo mismo que en regresión Ridge, es decir, el número de variables es demasido alto para poder interpretarlos, mientras que si escogemos el número de coefiecientes distintos de cero (`"df"`), solamente nos quedamos con 19, que ya es una cantidad razonable, teniendo en cuenta que algunas de ellas van a ser distintos niveles de las mismas variables factoriales.

4.3. Métodos de reducción de la dimensión

En esta sección aplicaremos las funciones descritas en el capítulo anterior en nuestra base de datos. En primer lugar realizaremos la regresión de componentes principales y, después, la regresión mínimo cuadrada parcial.

4.3.1. Componentes principales

El comando para realizar el análisis de componentes principales mencionado anteriormente es el siguiente:

```
> library(pls)
> pcreg <- pcr(SalePrice~Resto[,colSums(Resto !=0) > 2],data=datos,validation="CV",scale=T)
```

Vemos para poder utilizar esta función, hemos restringido el número de variables que incluímos en el modelo ya que, incluyéndolas todas, surgen problemas en el proceso de reescalado de la función. Hemos eliminado todas las variables que tengan menos de 2 valores. Nos podemos permitir este reducción del número de variables ya que teniendo cero o un solo valor no iba a influir demasiado en el resultado.

La primera componente principal tiene una varianza explicada del 3.96 %, mientras que la de la segunda es de solamente 1.93 %. Entre las 10 primeras componentes solamente suman el 15.30 % de la variabilidad explicada, y para alcanzar el 90 % tenemos que irnos a la componente 213 (línea roja en el Gráfico 4). Mediante validación cruzada, el número óptimo de componentes principales descendería a 142 (línea verde).

En cambio, para explicar el 90 % de la variabilidad de los precios de las viviendas (`SalePrice`), debemos utilizar las 259 primeras componentes principales (línea azul).

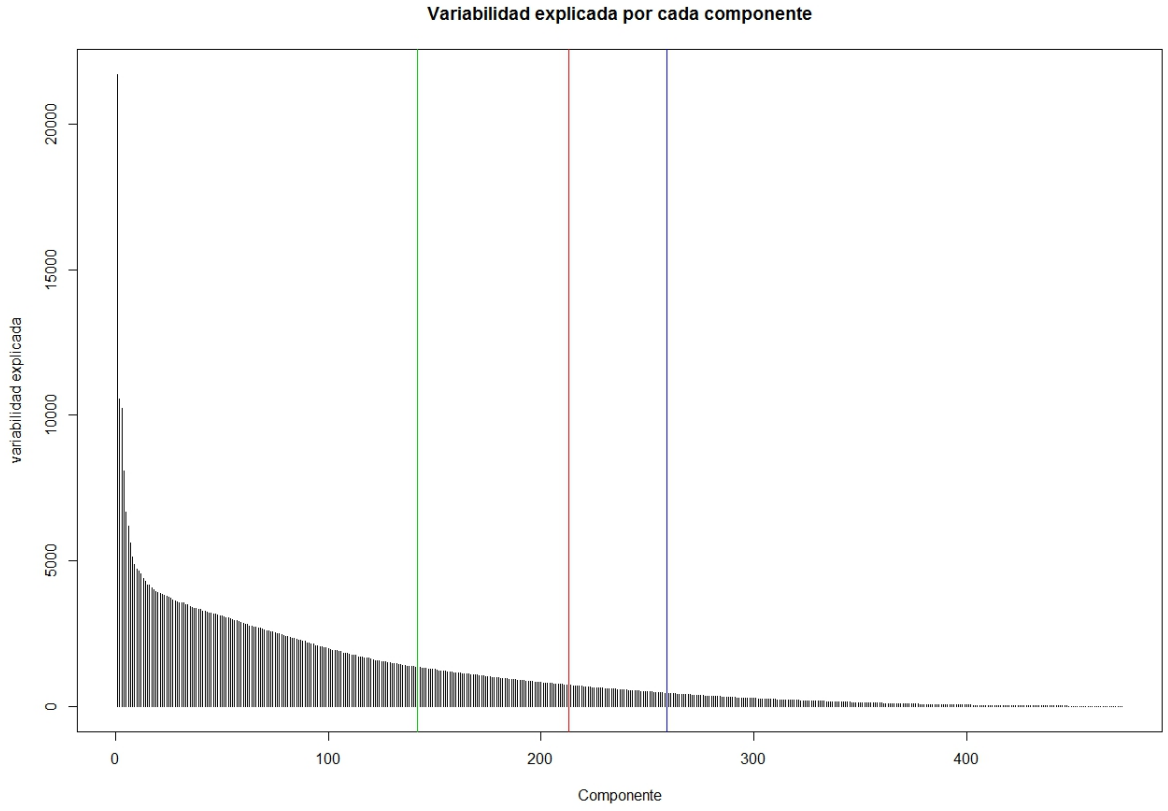

Gráfico 4: Variabilidad explicada por cada componente

4.3.2. Mínimos cuadrados parciales

En este apartado, utilizaremos la función `plsr()` para el caso de variable respuesta continua, y `plsRglm()` para el caso de variable respuesta binaria.

4.3.2.1. Variable respuesta continua

Utilizaremos la función `plsr()` para llevar a cabo la estimación.

```
> plsreg <- plsr(SalePrice~Resto[,colSums(Resto !=0) > 2],data=datos,validation="CV",scale=T)
```

Vemos que en este caso, también tuvimos que restringir las covariables que incorporamos al modelo. La primera componente principal tiene una varianza explicada del 3.82 %, con la que, a su vez, podríamos explicar el 71.29 % de la variabilidad de la variable respuesta `SalePrice`. Con solamente cinco componentes (línea azul en el Gráfico 5), ya explicamos el 90.68 % de la variabilidad de `SalePrice`. En cambio, para explicar el 90.01 % de la variabilidad de la matriz de covariables, necesitamos las 282 primeras componentes (línea roja). En cambio, por el método de validación cruzada, el número óptimo de componentes que necesitamos se reduce a solamente dos (línea verde).

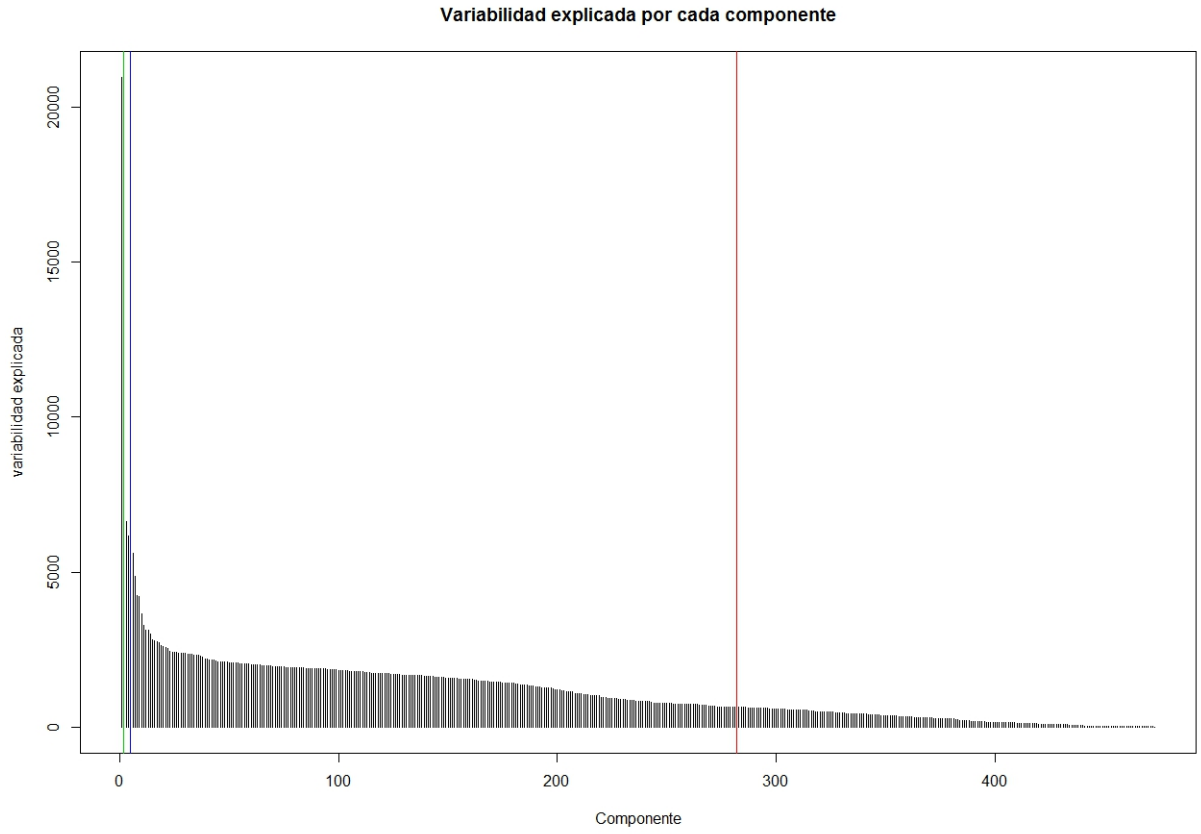

Gráfico 5: Variabilidad explicada por cada componente.

4.3.2.2. Variable respuesta binaria

Para llevar a cabo este método en R, utilizaremos la función de R `plsRglm()` anteriormente explicada. El comando utilizado es el siguiente:

```
> pls <- plsRglm(as.integer(Price)-1~., data=datos, nt=10, scaleX=T,
                 modele="pls-glm-logistic", MClassed=T, pvals.expli=T)
```

Vemos que hemos fijado el número máximo de componentes (`nt`) a 10. Es debido a que el tiempo de computación es muy elevado.

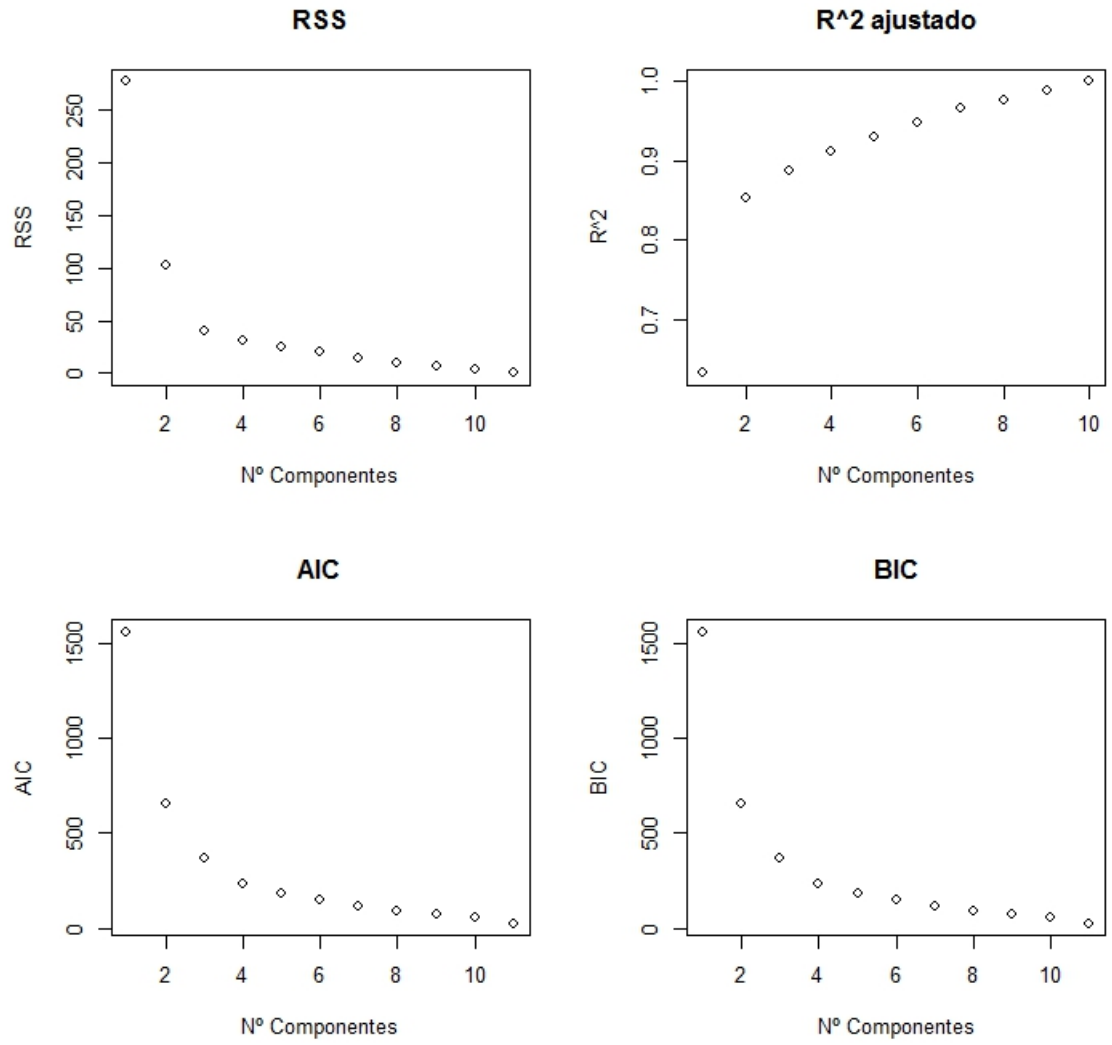

Gráfico 6: Criterios de selección para los modelos Partial Least Square con distinto número de componentes.

En el Gráfico 6 se muestran los distintos criterios de selección para el modelo PLS con distinto número de componentes. A pesar de que como máximo fijamos 10 componentes, en los gráficos de RSS, AIC y BIC aparecen 11 porque añaden el intercepto como una primera componente, cosa que no ocurre en el gráfico de R^2 ajustado.

Vemos que con los cuatro criterios, el mejor modelo es el que tiene el mayor número de componentes ya que los criterios dan un resultado menor, excepto en el caso del R^2 ajustado que es mayor. Vemos que en tres de los cuatro criterios (exceptuando el R^2 ajustado), a partir del modelo con tres componentes la reducción en el valor de los criterios se suaviza.

```
> coef(pls$FinalModel)

(Intercept)        tt.1        tt.2        tt.3        tt.4        tt.5
  -975.0390   1507.7811    782.5685    764.5819    876.2191   1011.5185
       tt.6        tt.7        tt.8        tt.9       tt.10
    650.0743   1110.1451    935.7702    600.1636    177.8313
```

4.4. Modelos Aditivos Generalizados

En este apartado, aplicaremos las funciones explicadas en el capítulo anterior para estimar modelos GAM. Estimaremos distintos modelos GAM, unos para el caso de variable respuesta continua, y otros para la variable respuesta binaria.

4.4.1. Variable respuesta continua

Para encontrar el modelo GAM final, hemos comenzado incluyendo todas las variables en el modelo y, escalonadamente, hemos eliminado aquellas que no eran significativas. Las variables continuas, en un principio, las hemos suavizado, mientras que las ordinales las hemos dejado sin suavizar. Cuando la suavización de las variables continuas no es significativa, las incluímos sin suavizar para comprobar si eran o no significativas, resultando que ninguna variable continua sin suavizar es significativa. La llamada a la función `gam()` que hemos realizado es la siguiente:

```
> gamc.final <- gam(SalePrice ~ s(Lot.Frontage, bs = "ps") + s(Lot.Area, bs = "ps") +
                    Street + Land.Contour + Lot.Config + Land.Slope + Neighborhood +
                    Condition.1 + Year.Built + Year.Remod.Add + Roof.Style +
                    Mas.Vnr.Type + s(Mas.Vnr.Area, bs = "ps") + Exter.Qual +
                    Bsmt.Qual + s(BsmtFin.SF.1, bs = "ps") + BsmtFin.Type.2 +
                    s(BsmtFin.SF.2, bs = "ps") + s(Bsmt.Unf.SF, bs = "ps") +
                    s(Total.Bsmt.SF, bs = "ps") + s(X1st.Flr.SF, bs = "ps") +
                    s(X2nd.Flr.SF, bs = "ps") + s(Low.Qual.Fin.SF, bs = "ps") +
                    s(Gr.Liv.Area, bs = "ps") + Bsmt.Full.Bath + TotRms.AbvGrd +
                    Functional + Fireplaces + Garage.Yr.Blt + Garage.Cars +
                    s(Wood.Deck.SF, bs = "ps") + s(Enclosed.Porch, bs = "ps") +
                    s(Screen.Porch, bs = "ps") + s(Pool.Area, bs = "ps") +
                    Mo.Sold + Yr.Sold + Sale.Type, data=datos)
```

Los resultados de las variables sin suavizar de la estimación son los siguientes:

```
> coef(gamc.final)

       (Intercept)
```

```
      -124076.1533

       StreetPave
       50245.5116

  Land.ContourHLS  Land.ContourLow  Land.ContourLvl
       20742.1030      -13052.6541       4181.0350

 Lot.ConfigCulDSac   Lot.ConfigFR2   Lot.ConfigFR3  Lot.ConfigInside
       11421.8574      -3491.0532     -23465.4844        3930.8142

    Land.SlopeMod    Land.SlopeSev
       -1011.6396      -68030.2979

NeighborhoodBlueste NeighborhoodBrDale NeighborhoodBrkSide NeighborhoodClearCr
        28009.8659        20800.2255         5965.6967        29669.2999
NeighborhoodCollgCr NeighborhoodCrawfor NeighborhoodEdwards NeighborhoodGilbert
         8772.7026        20688.3916        -7301.8882         6131.8051
 NeighborhoodGreens NeighborhoodIDOTRR NeighborhoodMeadowV NeighborhoodMitchel
        37745.7880         1253.2494        12334.5054        14043.6484
  NeighborhoodNAmes NeighborhoodNoRidge NeighborhoodNPkVill NeighborhoodNridgHt
        -2386.7197        14270.1601        19247.7518        25541.9646
 NeighborhoodNWAmes NeighborhoodOldTown  NeighborhoodSawyer NeighborhoodSawyerW
        11120.9901       -18627.8633         3293.8247         9284.3084
NeighborhoodSomerst NeighborhoodStoneBr   NeighborhoodSWISU  NeighborhoodTimber
        25967.6677        51720.4114        -7075.2884        12412.8437
NeighborhoodVeenker
        22384.2691

   Condition.1Feedr    Condition.1Norm     Condition.1PosA     Condition.1PosN
         9525.2074        22638.2796         42925.0124         16511.2498
    Condition.1RRAe    Condition.1RRAn     Condition.1RRNe     Condition.1RRNn
        22964.8305        14326.7032        -57433.5827         6621.8736

    Year.Built1880     Year.Built1882     Year.Built1890     Year.Built1892
       -23453.0389        20846.1032         -5598.8265        126447.7422
    Year.Built1893     Year.Built1900     Year.Built1904     Year.Built1905
        80856.5519        98646.3525         46302.5509         17766.8445
    Year.Built1908     Year.Built1910     Year.Built1913     Year.Built1914
        30589.7509        21216.9096        -24316.5617        116465.1577
    Year.Built1915     Year.Built1916     Year.Built1917     Year.Built1918
        23862.1264        36688.9800        -33322.7245         50755.6566
    Year.Built1919     Year.Built1920     Year.Built1921     Year.Built1922
        11723.6794       -20786.4620        -30810.7116          -865.6970
    Year.Built1923     Year.Built1924     Year.Built1925     Year.Built1926
        41044.9642        -2609.8149         18615.2883         68442.8893
    Year.Built1927     Year.Built1928     Year.Built1929     Year.Built1930
        66454.3262        19432.6261         17211.6335         38829.2797
```

Year.Built1931	Year.Built1932	Year.Built1934	Year.Built1935
58568.9850	-9783.6660	1560.0459	81541.7751
Year.Built1936	Year.Built1937	Year.Built1938	Year.Built1939
73526.5782	2930.8012	-6146.3305	-4768.5373
Year.Built1940	Year.Built1941	Year.Built1942	Year.Built1945
29445.9019	18391.8875	52893.7584	25689.1443
Year.Built1946	Year.Built1947	Year.Built1948	Year.Built1949
44557.0252	27255.2701	28360.3004	23875.0519
Year.Built1950	Year.Built1951	Year.Built1952	Year.Built1953
10451.0437	-98698.8290	-44433.0091	23356.6965
Year.Built1954	Year.Built1955	Year.Built1956	Year.Built1957
58560.2862	9197.2645	57788.0051	48993.8078
Year.Built1958	Year.Built1959	Year.Built1960	Year.Built1961
-4165.6702	56804.8596	-60125.3325	15484.2902
Year.Built1962	Year.Built1963	Year.Built1964	Year.Built1965
18273.0483	469.3139	4039.5289	41467.8703
Year.Built1966	Year.Built1967	Year.Built1968	Year.Built1969
65996.1784	-21999.8270	29671.4574	-15099.3281
Year.Built1970	Year.Built1971	Year.Built1972	Year.Built1973
101971.0227	29221.0388	-57055.5120	19394.5554
Year.Built1974	Year.Built1975	Year.Built1976	Year.Built1977
13784.1888	35062.7785	14070.9859	48717.3571
Year.Built1978	Year.Built1979	Year.Built1980	Year.Built1981
71403.0924	-418.0937	41201.1102	41979.4719
Year.Built1982	Year.Built1983	Year.Built1984	Year.Built1985
36649.2874	72982.6173	20187.6622	4784.3473
Year.Built1986	Year.Built1987	Year.Built1988	Year.Built1989
-70970.5725	13158.8126	11059.6746	56222.1699
Year.Built1990	Year.Built1991	Year.Built1992	Year.Built1993
-2222.8265	24895.9534	-22094.7026	69601.6064
Year.Built1994	Year.Built1995	Year.Built1996	Year.Built1997
5143.7636	-18417.8094	-3001.0470	-13932.2414
Year.Built1998	Year.Built1999	Year.Built2000	Year.Built2001
-13291.6979	124456.8164	119413.8697	124550.8747
Year.Built2002	Year.Built2003	Year.Built2004	Year.Built2005
55717.3886	40971.9585	69345.9426	47556.9125
Year.Built2006	Year.Built2007	Year.Built2008	Year.Built2009
70305.8490	83940.5703	98212.2085	103942.9228
Year.Built2010			
112087.7899			

Year.Remod.Add1951	Year.Remod.Add1952	Year.Remod.Add1953	Year.Remod.Add1954
4435.8813	64354.6903	23356.6965	-15226.4916
Year.Remod.Add1955	Year.Remod.Add1956	Year.Remod.Add1957	Year.Remod.Add1958
33509.2567	-4923.3573	-12461.6979	9570.3952
Year.Remod.Add1959	Year.Remod.Add1960	Year.Remod.Add1961	Year.Remod.Add1962
-12414.2426	84590.5497	-3150.7119	7974.5115
Year.Remod.Add1963	Year.Remod.Add1964	Year.Remod.Add1965	Year.Remod.Add1966

-14426.9526	-14198.3519	-38444.2636	-10893.7317
Year.Remod.Add1967	Year.Remod.Add1968	Year.Remod.Add1969	Year.Remod.Add1970
29934.2569	-7695.7953	15640.0355	-128306.5988
Year.Remod.Add1971	Year.Remod.Add1972	Year.Remod.Add1973	Year.Remod.Add1974
29221.0388	4911.5998	-2418.7016	-21063.6813
Year.Remod.Add1975	Year.Remod.Add1976	Year.Remod.Add1977	Year.Remod.Add1978
6163.1898	-1908.5222	-5829.1624	-7441.6588
Year.Remod.Add1979	Year.Remod.Add1980	Year.Remod.Add1981	Year.Remod.Add1982
-536.7306	-7765.0045	6009.8615	6793.2853
Year.Remod.Add1983	Year.Remod.Add1984	Year.Remod.Add1985	Year.Remod.Add1986
-23321.5071	29833.5792	10637.5118	13873.4631
Year.Remod.Add1987	Year.Remod.Add1988	Year.Remod.Add1989	Year.Remod.Add1990
-802.4005	4116.7915	-29345.0969	-2527.3657
Year.Remod.Add1991	Year.Remod.Add1992	Year.Remod.Add1993	Year.Remod.Add1994
-1214.6640	1804.6304	1009.5728	7160.7936
Year.Remod.Add1995	Year.Remod.Add1996	Year.Remod.Add1997	Year.Remod.Add1998
8573.7995	13385.1977	5992.4332	8384.3799
Year.Remod.Add1999	Year.Remod.Add2000	Year.Remod.Add2001	Year.Remod.Add2002
5535.2414	8851.8563	21710.5776	6399.6169
Year.Remod.Add2003	Year.Remod.Add2004	Year.Remod.Add2005	Year.Remod.Add2006
-5049.3346	13820.5450	17991.4342	19632.9592
Year.Remod.Add2007	Year.Remod.Add2008	Year.Remod.Add2009	Year.Remod.Add2010
32021.1442	17174.1277	18129.9437	32771.7571

Roof.StyleGable	Roof.StyleGambrel	Roof.StyleHip	Roof.StyleMansard
-27779.2754	-11328.9246	-26122.7968	-41880.7524
Roof.StyleShed			
71905.0410			

Mas.Vnr.TypeBrkFace	Mas.Vnr.TypeNone	Mas.Vnr.TypeStone	
-11490.6490	-35150.1884	-5846.2209	

Exter.QualFa	Exter.QualGd	Exter.QualTA	
-33343.1602	-28872.5318	-39661.6764	

Bsmt.QualFa	Bsmt.QualGd	Bsmt.QualTA	
1472.1503	-14741.3799	-11651.2208	

BsmtFin.Type.2ALQ	BsmtFin.Type.2BLQ	BsmtFin.Type.2GLQ	BsmtFin.Type.2LwQ
161391.3115	145057.4649	168382.1506	153430.4607
BsmtFin.Type.2Rec	BsmtFin.Type.2Unf		
147387.4570	154030.4641		

Bsmt.Full.Bath1	Bsmt.Full.Bath2		
8301.8720	15565.1060		

TotRms.AbvGrd4	TotRms.AbvGrd5	TotRms.AbvGrd6	TotRms.AbvGrd7
14953.8965	6689.5569	10496.3549	8831.1287

TotRms.AbvGrd8	TotRms.AbvGrd9	TotRms.AbvGrd10	TotRms.AbvGrd11
6539.6598	-393.5705	907.2181	5844.9412
TotRms.AbvGrd12	TotRms.AbvGrd15		
56728.5460	5446429.4468		
FunctionalMaj2	FunctionalMin1	FunctionalMin2	FunctionalMod
13120.0849	42341.1085	27226.4611	25372.6836
FunctionalTyp			
53992.0868			
Fireplaces2	Fireplaces3	Fireplaces4	
6386.6472	1429.5692	377914.7855	
Garage.Yr.Blt1908	Garage.Yr.Blt1910	Garage.Yr.Blt1914	Garage.Yr.Blt1915
30589.7509	76395.9453	116465.1577	81016.9204
Garage.Yr.Blt1916	Garage.Yr.Blt1917	Garage.Yr.Blt1918	Garage.Yr.Blt1919
36688.9800	120009.0896	11179.8213	85848.0569
Garage.Yr.Blt1920	Garage.Yr.Blt1921	Garage.Yr.Blt1922	Garage.Yr.Blt1923
119606.3310	162747.4651	73551.6410	51831.0313
Garage.Yr.Blt1924	Garage.Yr.Blt1925	Garage.Yr.Blt1926	Garage.Yr.Blt1927
79152.7546	90592.0705	28171.9216	66454.3262
Garage.Yr.Blt1928	Garage.Yr.Blt1929	Garage.Yr.Blt1930	Garage.Yr.Blt1931
95239.3721	88309.1352	66237.2357	38427.0029
Garage.Yr.Blt1932	Garage.Yr.Blt1934	Garage.Yr.Blt1935	Garage.Yr.Blt1936
153259.8032	115437.2483	29233.8911	47672.0008
Garage.Yr.Blt1937	Garage.Yr.Blt1938	Garage.Yr.Blt1939	Garage.Yr.Blt1940
110288.9782	105269.5213	97075.8275	68590.3942
Garage.Yr.Blt1941	Garage.Yr.Blt1942	Garage.Yr.Blt1943	Garage.Yr.Blt1945
72652.0920	52893.7584	120677.7023	73765.7587
Garage.Yr.Blt1946	Garage.Yr.Blt1947	Garage.Yr.Blt1948	Garage.Yr.Blt1949
50540.5255	75443.6677	69177.6862	73023.2551
Garage.Yr.Blt1950	Garage.Yr.Blt1951	Garage.Yr.Blt1952	Garage.Yr.Blt1953
90234.0137	173091.9113	64354.6903	23356.6965
Garage.Yr.Blt1954	Garage.Yr.Blt1955	Garage.Yr.Blt1956	Garage.Yr.Blt1957
45431.6813	59953.4086	28720.3189	48993.8078
Garage.Yr.Blt1958	Garage.Yr.Blt1959	Garage.Yr.Blt1960	Garage.Yr.Blt1961
87275.0385	51293.0618	65234.6619	80063.1665
Garage.Yr.Blt1962	Garage.Yr.Blt1963	Garage.Yr.Blt1964	Garage.Yr.Blt1965
72811.2384	93165.0660	103961.8248	75018.4712
Garage.Yr.Blt1966	Garage.Yr.Blt1967	Garage.Yr.Blt1968	Garage.Yr.Blt1969
39745.5060	93435.7851	62079.0911	94737.0545
Garage.Yr.Blt1970	Garage.Yr.Blt1971	Garage.Yr.Blt1972	Garage.Yr.Blt1973
97189.5974	29221.0388	148424.9275	51784.6182
Garage.Yr.Blt1974	Garage.Yr.Blt1975	Garage.Yr.Blt1976	Garage.Yr.Blt1977
93620.7023	57375.9160	80947.2746	40454.2740
Garage.Yr.Blt1978	Garage.Yr.Blt1979	Garage.Yr.Blt1980	Garage.Yr.Blt1981
22110.6713	100536.7579	52737.5179	50409.6956
Garage.Yr.Blt1982	Garage.Yr.Blt1983	Garage.Yr.Blt1984	Garage.Yr.Blt1985

```
       36649.2874         85487.2003         65992.5564         76443.7117
Garage.Yr.Blt1986  Garage.Yr.Blt1987  Garage.Yr.Blt1988  Garage.Yr.Blt1989
      169631.3501        117442.6519         89155.7772         56222.1699
Garage.Yr.Blt1990  Garage.Yr.Blt1991  Garage.Yr.Blt1992  Garage.Yr.Blt1993
       90226.8927         74314.6025        108813.0186         36919.3365
Garage.Yr.Blt1994  Garage.Yr.Blt1995  Garage.Yr.Blt1996  Garage.Yr.Blt1997
      101989.6164        126047.6386        108577.0916        125539.3108
Garage.Yr.Blt1998  Garage.Yr.Blt1999  Garage.Yr.Blt2000  Garage.Yr.Blt2001
      127415.5016         -2386.1453        -10093.1397        -12066.1579
Garage.Yr.Blt2002  Garage.Yr.Blt2003  Garage.Yr.Blt2004  Garage.Yr.Blt2005
       55717.3886         74852.2215         35144.7012         55223.9729
Garage.Yr.Blt2006  Garage.Yr.Blt2007  Garage.Yr.Blt2008  Garage.Yr.Blt2009
       31348.6239          7473.6682         25396.9725         24162.9376
Garage.Yr.Blt2010
      -14885.9898

     Garage.Cars2       Garage.Cars3       Garage.Cars4
        6798.8305         26048.2639         66177.1859

        Mo.Sold2           Mo.Sold3           Mo.Sold4           Mo.Sold5
       -8535.7107        -10708.4558         -9794.1061         -6429.5407
        Mo.Sold6           Mo.Sold7           Mo.Sold8           Mo.Sold9
       -8839.8759         -6842.1855        -11015.3604         -5433.3014
       Mo.Sold10          Mo.Sold11          Mo.Sold12
       -8485.4406         -9538.2040         -8450.4081

      Yr.Sold2007        Yr.Sold2008        Yr.Sold2009        Yr.Sold2010
       -2590.8313         -2714.3708         -7380.5622         -3049.4024

     Sale.TypeCon     Sale.TypeConLD     Sale.TypeConLI     Sale.TypeConLw
       49368.5166         13028.1969         25531.4700         25383.5428
     Sale.TypeCWD       Sale.TypeNew       Sale.TypeVWD       Sale.TypeWD
       19185.3885         23655.3108          3057.7230         14892.5295
```

El factor `Pave` de la variable `Street` es positivo, es decir, cuando pasamos de que la carretera de acceso a la vivienda no esté pavimentada a que sí lo esté, se incrementa el precio de la vivienda en 50,245.51$.

En cuanto a la variable `Land.Contour` (llanura de la propiedad), vemos que, con respecto al factor de referencia (`Bnk`, está significativamente más elevado que el nivel de la calle), que la vivienda esté por debajo del nivel de la calle (`Low`) reduce el precio de la vivienda. En cambio, si la vivienda está en una zona en cuesta por encima del nivel de la calle (`HLS`) o casi a nivel de la calle (`Lvl`), el precio de la vivienda aumenta.

El factor de referencia que hemos tomado para la variable `Lot.Config` es `Corner` (el terreno hace esquina con dos calles). Cuando pasamos a que el terreno esté situado con dos fachadas

hacia varias calles (FR2) o que tres de las fachadas den a varias calles (FR3), el efecto sobre el precio es negativo, reduciendo éste en 3,491.05 y 23.465.49$, respectivamente. En cambio, si pasamos a que la vivienda esté situada en una calle sin salida (CulDSac) o que de solamente a una calle (Inside), el precio se incrementa en 11,421.86 y 3,930.81$, respectivamente.

En cuanto a la inclinación del terreno, el factor de referencia es que está suavemente inclinado (Gtl), una vez cambiamos de factor, siempre tiene un efecto negativo en el precio, como era de esperar. El precio se reduce en 1,011.64$ cuando el terreno está moderadamente inclinado (Mod) y, cuando está fuertemente inclinado (Sev), se reduce el precio en 68,030.30$.

De los vecindarios, el que utilizamos como referencia es el de Bloomington Heights (Blmngtn). Con respecto a este, hay algunos que incrementan el precio y otros que lo disminuyen. El que más incrementa el precio, es decir, el que podríamos considerar como el barrio más caro de los participantes en el estudio, es el de Stone Brook (StoneBr), que incrementa el precio en 51,720.41$. En cambio, el que lo disminuye en mayor medida o barrio más barato es el de Old Town, que disminuye el precio en 18,627.86$.

También podemos ver que el precio de las viviendas varía según lo cercano que tengamos algunos servicios (Condition.1). El factor de referencia es que nuestro terreno se encuentre adyacente a una de las calles principales de la ciudad (Artery). Con respecto a este factor, solamente existe una condición que reduzca el precio, que es vivir a menos de 200 metros de la estación de ferrocarril de East-West (RRNe), que lo reduce en 57,433.58$. Del resto de condiciones, la que más incrementa el precio es estar junto a lo que podríamos denominar como *zonas positivas* (PosA), como pueden ser parques o zonas naturales. En este caso, el incremento en el precio es de 42,925.01$. En cambio, poder residir cerca de estas zonas positivas (PosN), solamente incrementa el precio en 16,511.25$.

Con respecto al año de construcción de la vivienda (Year.Built), el año de referencia es el de la vivienda más antigua vendida durante los años de estudio, que es de 1879. Al cambiar de año, hay algunos que incrementan el precio y otros que lo reducen, siendo el año que más lo reduce 1951, en 98,698.83$, y el que más lo incrementa el año 1892 en 126,447.74$. Aunque cabe destacar que hasta el año 1990, tenemos como mucho cinco viviendas construidas en cada año, por lo que son muy pocos datos para cada año como para poder sacar conclusiones reales.

Con la variable Year.Remod.Add (año de remodelación de la vivienda, en caso de no haber sido remodelada, este año coincidirá con el de contrucción, Year.Built) sucede algo parecido que con el año de construcción. El año de referencia es 1950, y vemos que algunos años incrementan el precio mientras otros lo reducen. El año que más reduce el precio es 1970, en 128,306.60$. En cambio, el que más lo incrementa es 1960 en 84,590.55$. Pero igual que con la variable Year.Built, hay muy pocas viviendas remodeladas cada año.

La variable Roof.Style (estilo del tejado) también influye sobre el precio de las viviendas. El factor de referencia es que el tejado sea plano (Flat). Cuando pasamos de éste a otro tipo de

tejado, solamente hay uno que incremente el precio de la vivienda, que es el tipo `Shed` (tejado a una vertiente, es decir, el tejado tiene solamente una parte inclinada que ocupa el total de la vivienda), con el que el precio se incrementaría en 71,905.04$. El resto reducen el precio de la vivienda, siendo el que más lo reduce el tipo `Mansard` (tejado de cuatro cubiertas cuya parte central es un tejado normal pero el resto tiene un pendiente mucho más pronunciada) en 41,880.75$; el siguiente el estilo `Gable` (es el estilo habitual de dos vertientes), reduciéndolo en 27,779.27$; el siguiente el tipo `Hip` (tejado de cuatro cubiertas) en 26,122.80$; y, por último, el que menos reduce el precio es el tipo `Gambrel` (es igual que el tejado Mansard pero con solamente dos cubiertas), que lo reduce en 11,328.92$.

Con respecto a los materiales utilizados en el acabado de los muros de las viviendas (`Mas.Vnr.Type`), el factor que utilizamos de referencia es `BrkCmn` (ladrillo). Con respecto a este material, el resto de materiales utilizados reducen el precio de la vivienda. El que más lo disminuye es que los muros no tengan ningún acabo (`None`); el siguiente que más reduce el precio es que los muros sean de ladrillo a la vista (`BrkFace`); y, por último, lo que menos reduce los precios de las viviendas son los muros de piedra `Stone`.

En relación a la variable anterior, tenemos la variable que evalúa la calidad de los materiales del exterior (`Exter.Qual`). Esta variable tiene cinco niveles: `Ex` (excelente), `Gd` (bueno), `TA` (en la media), `Fa` (aceptable) y `Po` (pobre). No hay ninguna vivienda que tenga la última calificación. Se toma como referencia el excelente, y vemos que cualquier otra calificación va a reducir el precio de la vivienda, como era de esperar.

La variable `Bsmt.Qual` evalúa el tamaño del sótano. Tiene las mismas clasificaciones que la variable anterior, pero en este caso cada cualificación implica un tamaño de sótano: `Ex` (Excellent, más de 100 pulgadas), `Gd` (Good, 90-99 pulgadas), `TA` (Typical, 80-89 pulgadas), `Fa` (Fair, 70-79 pulgadas) y `Po` (Poor, menos de 70 pulgadas). La cualificación de referencia es excelente, por lo que, como podemos ver en los resultado de la estimación, que la vivienda tenga otra cualificación reduce el precio, exceptuando pasar a la cualificación `Fa`, que lo incrementa. Que `Fa` incremente el precio de la vivienda carece de sentido, de hecho, ese factor no pasa el test de significación en nuestro análisis. Que obtengamos ese extraño resultado se puede deber a que solamente tenemos 15 viviendas con esa calificación.

La variable `BsmtFin.Type.1` también evalúa el sótano, pero en este caso en términos de si podría ser utilizado como vivienda o no, pero esta variable no resultó significativa en nuestro análisis, pero sí lo resultó la variable `BsmtFin.Type.2`, que se utiliza para el caso de que una sola cualidad no describa totalmente el sótano. En este caso, el factor de referencia es que la vivienda no tenga sótano, por eso, cualquier otra cualificación incrementa el precio. Cabe destacar que en la clase `Unf` (sótano sin terminar) se encuentran casi todas las viviendas, 1024 de 1159.

Las siguiente variable, `Bsmt.Full.Bath`, indican el número de baños completos que hay en el sótano. El número de baños de referencia es cero y, de hecho, es el número que más se da con 605 de 1159. Vemos que, como era de esperar, según aumentamos el número de baños se incrementa

el precio de la vivienda.

La variable `TotRms.AbsGrd` tiene en cuenta todas las habitaciones que están por encima del nivel de la calle, exceptuando los baños. El número de habitaciones de referencia es 3 y, podemos observar que, exceptuando con 9 habitación, todas incrementan el precio de la vivienda. Que con 9 habitaciones se reduzca el precio es extraño, de hecho, ese factor resultó no significativo para nuestro modelo.

La siguiente variable indica la funcionalidad de la vivienda (`Functional`). El nivel más alto de esta variable es `Typ`, que significa funcionalidad habitual, y el resto de niveles van en función de la pérdida de funcionalidad.El nivel de referencia es `Maj1` (grandes reducciones en la funcionalidad 1) y, solamente se encuentra por debajo el nivel `Maj2` (grandes reducciones en la funcionalidad 1). Por eso, solamente este factor debería tener un efecto negativa en la estimación, pero vemos que su efecto es positivo. Esto se puede deber a que solamente hay dos viviendas que se encuentran en el factor `Maj2`, por lo que no se puede estimar correctamente el efecto y, de hecho, no resulta significativo en nuestro análisis.

En el caso de la variable `Fireplaces` (número de chimeneas en la vivienda), cabe destacar que solamente el factor de dos chimeneas resultó significativo, ya que con tres y cuatro chimeneas solamente tenemos nueve y una viviendas, respectivamente. El factor de referencia es tener una sola chimenea, por lo que vemos que el efecto de tener dos, como era de esperar, incrementa el precio de la vivienda en 6,386.65$.

Con respecto al año de construcción del garage (`Garage.Yr.Blt`), el año de referencia es 1900. Con respecto a este año, solamente hay cuatro que reduzcan el precio, que son 1999, 2000, 2001 y 2010, incrementando la reducción en el precio según avanzan los años. Cabe destacar que estos cuatro años no resultaron significativos en nuestro modelo. En cambio, el año que más incrementa el precio de la vivienda es 1951, que lo incrementa en 173,091.91$.

En cuanto al número de coches que se pueden aparcar en el garage (`Garage.Cars`), el número de referencia es uno, y vemos que, según aumenta el número de coches, se incrementa también el precio de la vivienda, llegando a aumentar en 66,177.19$.

La fecha en la que se vendió la vivienda (el mes, `Mo.Sold`, y el año,`Yr.Sold`) también influye en el precio. Vemos que el mes más caro es el de referencia, Enero, ya que todos los meses restantes reducen el precio de la vivienda, siendo el mes más caro Agosto. En cuanto al año, también ocurre que el año más caro es el de referencia, 2006, siendo el más caro el 2009.

Por último, vemos que el tipo de venta también influye en el precio. El factor de referencia es `COD` (la venta es realizada por un fiduciario, que puede ser un administrador del estado) y, a partir de este, el resto de tipos de ventas incrementan el precio de la vivienda, siendo el que más lo incrementa `Con` (venta mediante un contrato con un pago inicial del 15 % con respecto al valor de la vivienda).

A continuación, presentamos los resultados de la estimación de las variables suavizadas.

```
                    edf Ref.df      F  p-value
s(Lot.Frontage)  1.0000 1.0000 10.685 0.001128 **
s(Lot.Area)      1.1625 1.3000 38.208 4.79e-11 ***
s(Mas.Vnr.Area)  7.5683 7.9432  4.375 3.78e-05 ***
s(BsmtFin.SF.1)  4.9789 5.1642  9.183 1.23e-08 ***
s(BsmtFin.SF.2)  2.1987 2.6971  4.778 0.004174 **
s(Bsmt.Unf.SF)   1.1597 1.3394  6.102 0.008431 **
s(Total.Bsmt.SF) 2.6416 3.1310 10.929 3.60e-07 ***
s(X1st.Flr.SF)   5.6925 5.7935 23.051  < 2e-16 ***
s(X2nd.Flr.SF)   4.0408 4.7911  6.517 1.04e-05 ***
s(Low.Qual.Fin.SF) 0.9655 0.9655  9.145 0.003056 **
s(Gr.Liv.Area)   5.5952 6.0622 12.574 1.15e-13 ***
s(Wood.Deck.SF)  4.9646 5.6678  4.211 0.000522 ***
s(Enclosed.Porch) 2.2414 2.7279  4.254 0.007612 **
s(Pool.Area)     5.7069 5.9453  3.249 0.003845 **
s(Screen.Porch)  1.0000 1.0000 10.956 0.000977 ***
```

Con respecto a las variables suavizadas del modelo, vemos que todas tienen un `p-value` menor que 0.05, por lo que rechazamos la hipótesis nula de que sus coeficientes sean iguales a cero con un nivel de significación del 5 %. Es decir, todas las suavizaciones son significativas. Vemos que las variables `Lot.Frontage` (metros de calle conectados con la propiedad), `Lot.Area` (tamaño del terreno en pies cuadrados), `Bsmt.Unf.SF` (pies cuadrados de sótano sin terminar), `Low.Qual.Fin.SF` (terminado de baja calidad en pies cuadrados en todas las plantas de la vivienda) y `Screen.Porch` (tamaño de porche semicerrado medido en pies cuadrados) tienen un grado de libertad (`edf`), o aproximadamente uno, lo que quiere decir que sus funciones suavizadas son rectas. En el Gráfico 7, podemos comprobar que lo son. En cambio, el resto de variables son curvas, siendo la correspondiente a la variable `Mas.Vnr.Area` (pies cuadrados de materiales de alta calidad utilizados en los muros de la vivienda) la que tiene el mayor número de grados de libertad con 7.57. Podemos comprobar en el Gráfico 7, que la curva correspondiente a esta variable es la más complicada ya que, según aumentamos el número de grados de libertad, la curva suavizada se vuelve más compleja.

En ese mismo gráfico, podemos obserbar que las variables `Lot.Frontage`, `Lot.Area`, `Total.Bsmt.SF` (tamaño del sótano en pies cuadrados), `X2nd.Flr.SF` (pies cuadrados del segundo piso), `Low.Qual.Fin.SF`, `Enclosed.Porch` (pies cuadrados de porche cerrado) y `Screen.Porch`, tienen un efecto positivo sobre el precio de las viviendas (`SalePrice`), es decir, según se incrementa el valor de estas variables, aumenta el precio de las viviendas. Esta situación tiene sentido para todas las variables excepto para `Low.Qual.Fin.SF`, ya que al incrementarse el área de terminados de baja calidad en la vivienda, no debería incrementarse el precio, sino reducirse. Pero este efecto puede estar provocado por el reducido número de datos que son mayor que cero. Solamente 11 de las 1159 viviendas tienen un valor distinto de cero en esta variable, por lo que no podemos estimar claramente su efecto.

En cambio, las variables `BsmtFin.SF.2` (se refiere a que cuando se valora el estado del sótano, puede tener dos características a la vez por lo que la variable `BsmtFin.SF.1` se refiere a los pies cuadrados del sótano que cumplen la primera calificación, mientras que la variable `BsmtFin.SF.2` se refiere a los pies cuadrados del sótano que cumplen la segunda calificación, en caso de que exista), `Bsmt.Unf.SF` y `Gr.Liv.Area` (pies cuadrados habitables en la vivienda que están por encima del nivel del terreno), tienen un efecto negativo sobre el precio. Es decir, según aumenta el valor de estas variables, se reduce el precio de las viviendas. De estas tres variables, solamente `Bsmt.Unf.SF` tiene sentido que tenga efecto negativo sobre el precio ya que, cuando aumentan los pies cuadrados de sótano sin termina, es lógico que disminuya el precio. Pero que al incrementar el número de pies cuadrados habitables en las plantas superiores de la vivienda se reduzca el precio va en contra de toda lógica.

En cuanto al resto de variables, vemos que no tienen el mismo efecto a lo largo de todos los valores que toma la variable. En el caso de la variable `Mas.Vnr.Area`, vemos que para los primeros valores el precio disminuye, hasta que, llegando a los 400 pies cuadrados, el precio se recupera. Cerca de los 500 pies cuadrados, se pierde el efecto ya que vemos que el valor cero se encuentra dentro de los intervalos de confianza (zonas sombreadas). En los valores más altos de la variable, vemos que hay una gran subida, pero no es muy fiable ya que el número de datos en esos valores es muy reducido.

En el caso de la variable `BsmtFin.SF.1`, según aumentan sus valores el precio se reduce, hasta que, llegados a los 2000 pies cuadrados, este efecto se estabiliza. En cambio, con la variable `X1st.Flr.SF` (pies cuadrados de la planta baja) ocurre lo contrario, a medida que aumenta el tamaño de la planta, aumenta también el precio de la vivienda.

La variable `Wood.Deck.SF` (tamaño del porche de madera medido en pies cuadrados) tiene un comportamiento oscilante. En un comienzo, no tiene efecto hasta que aproximadamente a los 200 pies cuadrados tiene un efecto positivo, es decir, a medida que aumenta el tamaño del porche de madera, se incrementa el precio de la vivienda. A partir de los 400 pies cuadrados, el efecto pasa a ser negativo reduciendo el precio y, apartir de los 600 pies, el precio vuelve a incrementarse, aunque no se puede saber con seguridad su comportamiento ya que los intervalo se amplian demasiado y, además, contienen al cero, por lo que existe la posibilidad de que para esos tamaños, no tenga ningún efecto sobre el precio.

La variable `Enclosed.Porch` comienza sin ser significativa ya que contiene el cero en su intervalo, hasta que a partir de los 200 pies cuadrados comienza a tener un efecto positivo. Es decir, a medida que se incrementa la dimensión del porche, aumenta también el precio de la vivienda.

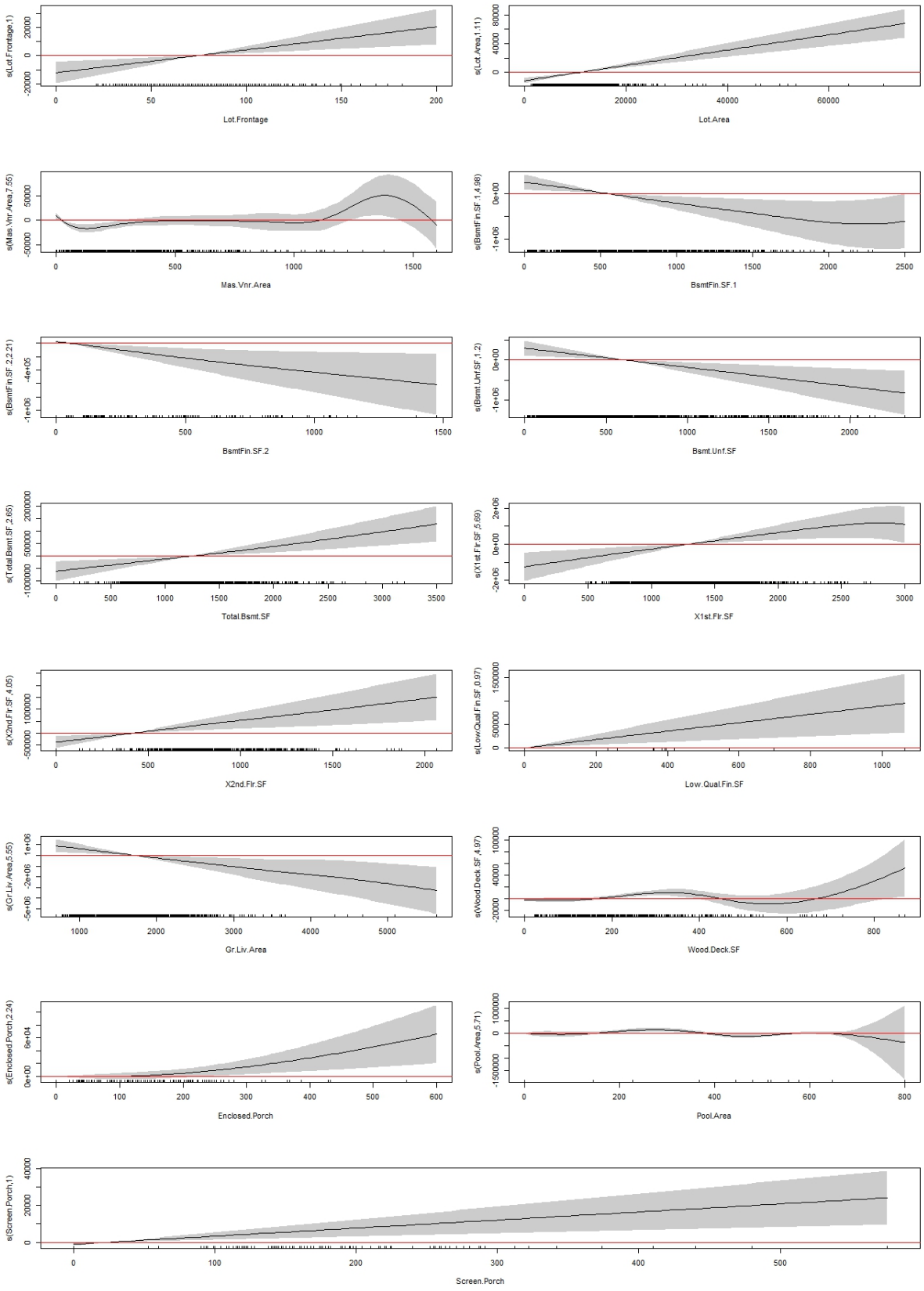

Gráfico 7: Funciones suavizadas de las variables continuas del modelo con los intervalos de confianza sombreados.

La variable `Pool.Area` (tamaño de la zona de la piscina medida en pies cuadrados) tienen muy pocos valores distintos de cero, solamente 11 de 1159. Por eso es una curva muy oscilante y que, en general, incluye el cero en su intervalo de confianza, por lo que no es muy fiable su estimación.

4.4.2. Variable respuesta binaria

El proceso que realizamos para llegar al modelo final consistió en, comenzando con todas las covariables, ir eliminando progresivamente las que no resultaban significativas, resultando así cada uno de los cuatro modelos que se comparan. Una vez finalizado el proceso, la llamada a la función `gam()` que realizamos para estimar el modelo final es la siguiente:

```
> gam.final <- gam(Price ~ s(Lot.Area, bs = "ps") + Land.Contour + Neighborhood +
                   Year.Remod.Add + Bsmt.Qual + s(BsmtFin.SF.2, bs = "ps") +
                   s(Bsmt.Unf.SF, bs = "ps") + s(X1st.Flr.SF, bs = "ps") + s(X2nd.Flr.SF,
                   bs = "ps") + Bsmt.Full.Bath + s(Wood.Deck.SF, bs = "ps") +
                   s(Screen.Porch, bs = "ps"), data=datos, family="binomial")
```

Las variables de tipo factor las incluimos en el modelo directamente, mientras que a las variables continuas les aplicamos suavización. En este caso, al especificar `bs = "ps"` dentro de cada suavización, le estamos indicando a la función que aplique una base de p-splines.

A continuación presentamos las odd-ratio de cada uno de los factores:

```
> exp(coef(gam.final))

                (Intercept)
                1.068978e-07

            Land.ContourHLS     Land.ContourLow     Land.ContourLvl
                1.433704e+06        1.670526e+05        5.153313e+04

       NeighborhoodBlueste  NeighborhoodBrDale NeighborhoodBrkSide NeighborhoodClearCr
                1.068118e-29        7.024507e-21        1.082791e+01        4.940797e-02
      NeighborhoodCollgCr NeighborhoodCrawfor NeighborhoodEdwards NeighborhoodGilbert
                6.935261e-01        6.681056e+00        4.716930e-02        4.189508e-01
       NeighborhoodGreens  NeighborhoodIDOTRR NeighborhoodMeadowV NeighborhoodMitchel
                2.946499e-32        8.137904e-31        1.316179e-28        4.350586e+00
        NeighborhoodNAmes NeighborhoodNoRidge NeighborhoodNPkVill NeighborhoodNridgHt
                1.956371e-03        3.034127e+00        4.688959e-26        2.289593e+00
       NeighborhoodNWAmes NeighborhoodOldTown  NeighborhoodSawyer NeighborhoodSawyerW
```

```
       2.544625e-03        5.162742e-06        1.548033e-35        2.436361e-01
NeighborhoodSomerst NeighborhoodStoneBr   NeighborhoodSWISU  NeighborhoodTimber
       3.803753e+00        4.556155e+00        2.344656e-39        1.097100e+00
NeighborhoodVeenker
       2.349537e-01

Year.Remod.Add1951  Year.Remod.Add1952  Year.Remod.Add1953  Year.Remod.Add1954
      3.815658e-25        4.277931e-32        7.006671e+00        3.703399e+04
Year.Remod.Add1955  Year.Remod.Add1956  Year.Remod.Add1957  Year.Remod.Add1958
      7.091431e-20        6.055820e-01        1.625007e-30        6.657231e-27
Year.Remod.Add1959  Year.Remod.Add1960  Year.Remod.Add1961  Year.Remod.Add1962
      1.305635e-32        1.094496e-32        1.748272e-27        3.231488e-26
Year.Remod.Add1963  Year.Remod.Add1964  Year.Remod.Add1965  Year.Remod.Add1966
      7.898143e-05        4.076812e-29        1.568598e-14        2.730998e+03
Year.Remod.Add1967  Year.Remod.Add1968  Year.Remod.Add1969  Year.Remod.Add1970
      4.677246e-16        4.233881e+00        2.681560e-32        2.979885e+01
Year.Remod.Add1971  Year.Remod.Add1972  Year.Remod.Add1973  Year.Remod.Add1974
      2.309424e-30        3.753092e-24        1.638995e+02        3.018488e-01
Year.Remod.Add1975  Year.Remod.Add1976  Year.Remod.Add1977  Year.Remod.Add1978
      3.410260e-03        2.371229e-05        1.426416e-29        2.243236e+02
Year.Remod.Add1979  Year.Remod.Add1980  Year.Remod.Add1981  Year.Remod.Add1982
      6.916826e+01        1.449694e+02        1.727182e+03        4.849953e-32
Year.Remod.Add1983  Year.Remod.Add1984  Year.Remod.Add1985  Year.Remod.Add1986
      1.022960e-24        3.524664e+01        1.888687e+01        2.767137e+02
Year.Remod.Add1987  Year.Remod.Add1988  Year.Remod.Add1989  Year.Remod.Add1990
      3.006713e+03        2.001548e+02        1.236524e+00        1.589234e+00
Year.Remod.Add1991  Year.Remod.Add1992  Year.Remod.Add1993  Year.Remod.Add1994
      6.838498e+00        3.148179e+00        5.877145e+01        2.221393e+01
Year.Remod.Add1995  Year.Remod.Add1996  Year.Remod.Add1997  Year.Remod.Add1998
      8.228281e+01        2.517145e+02        6.893542e+02        1.774223e+01
Year.Remod.Add1999  Year.Remod.Add2000  Year.Remod.Add2001  Year.Remod.Add2002
      1.308643e+02        1.238111e+02        6.328655e+01        5.382929e+01
Year.Remod.Add2003  Year.Remod.Add2004  Year.Remod.Add2005  Year.Remod.Add2006
      1.488987e+01        2.180040e+02        2.938481e+02        1.786715e+02
Year.Remod.Add2007  Year.Remod.Add2008  Year.Remod.Add2009  Year.Remod.Add2010
      5.256475e+02        3.928213e+02        5.271788e+06        8.386365e+36

       Bsmt.QualFa         Bsmt.QualGd         Bsmt.QualTA
      1.019332e+01        1.244657e-01        9.980651e-02

   Bsmt.Full.Bath1     Bsmt.Full.Bath2
      1.253730e+01        3.105287e+02
```

Vemos que, por ejemplo, con la variable `Land.Contour` (llanura de la propiedad) ocurre que, si pasamos de su factor de referencia (`Bnk`, la propiedad está significativamente más elevado que el nivel de la calle), a cualquier otro factor, implica que la probabilidad de que el precio de la vivienda se encuentra por encima de la media de las viviendas se incrementa. Siendo el aumento

de esta probabilidad mayor en caso de que el terreno se encuentre significativamente inclinado (`HLS`). A continuación el que más incrementa esta probabilidad es que el terreno se encuentre en una depresión (`Low`) y, por último, el que menos lo aumenta es que se encuentre a nivel del suelo (`Lvl`).

En cuanto a los barrios, el que incrementaría en mayor medida la probabilidad de que el precio de la vivienda esté por encima de la media es el de Brookside (`BrkSide`), mientras que el que más reduciría esa probabilidad es el de South & West of Iowa State University (`SWISU`).

El año de remodelación de la vivienda que más incrementa la probabilidad de que la vivienda se encuentra en el grupo que está por encima de la media es el 2010, mientras que el que más reduce esta probabilidad es 1982. Cabe destacar que, en general, los años que están por encima de 1978, incrementan esta probabilidad de tener un precio por encima de la media, mientras que los años que están por debajo, la reducen.

Con respecto al tamaño del sótano (`Bsmt.Qual`), vemos que pasar del factor de referencia `Ex` (más de 100 pulgadas) al factor `Fa` (entre 70-79 pulgadas) incrementa la probabilidad de que el precio esté por encima de la media. En cambio, que el tamaño del sótano pase a estar entre 80 y 89 pulgadas (`TA`) o entre 90 y 99 pulgadas (`Gd`), disminuye esta probabilidad. A pesar de que carezca de sentido que al reducirse el tamaño se incremente la probabilidad de que el precio de la vivienda esté por encima de la media, se puede deber a que solamente tenemos 15 viviendas en esta situación, por lo que la estimación del coeficiente de esta variable no es muy fiable.

Por último, al incrementarse el número de baños completos en el sótano (`Bsmt.Full.Bath`), se incrementa también la probabilidad de que el precio de la vivienda esté por encima de la media.

Vemos a continuación que todas las variables suavizadas son significativas con un nivel de significación de, al menos, un 5 %. En el caso de la variable `Lot.Area` (pies cuadrados de la propiedad), vemos que tiene un solo grado de libertad (`edf`), por lo que vamos a poder representar su curva suavizada con una recta. En cambio, el resto de variables tienen un mayor número de grados de libertad por lo que no es tan fácil representar su curva suavizada.

```
                edf Ref.df Chi.sq  p-value
s(Lot.Area)   1.000  1.000  17.74 2.53e-05 ***
s(BsmtFin.SF.2) 7.917  7.993  15.75  0.04594 *
s(Bsmt.Unf.SF)  7.449  7.780  21.37  0.00558 **
s(X1st.Flr.SF)  4.058  4.299  59.91 8.29e-12 ***
s(X2nd.Flr.SF)  4.296  4.710  62.19 5.62e-12 ***
s(Wood.Deck.SF) 3.083  3.628  12.35  0.01161 *
s(Screen.Porch) 4.631  5.085  14.03  0.01646 *
```

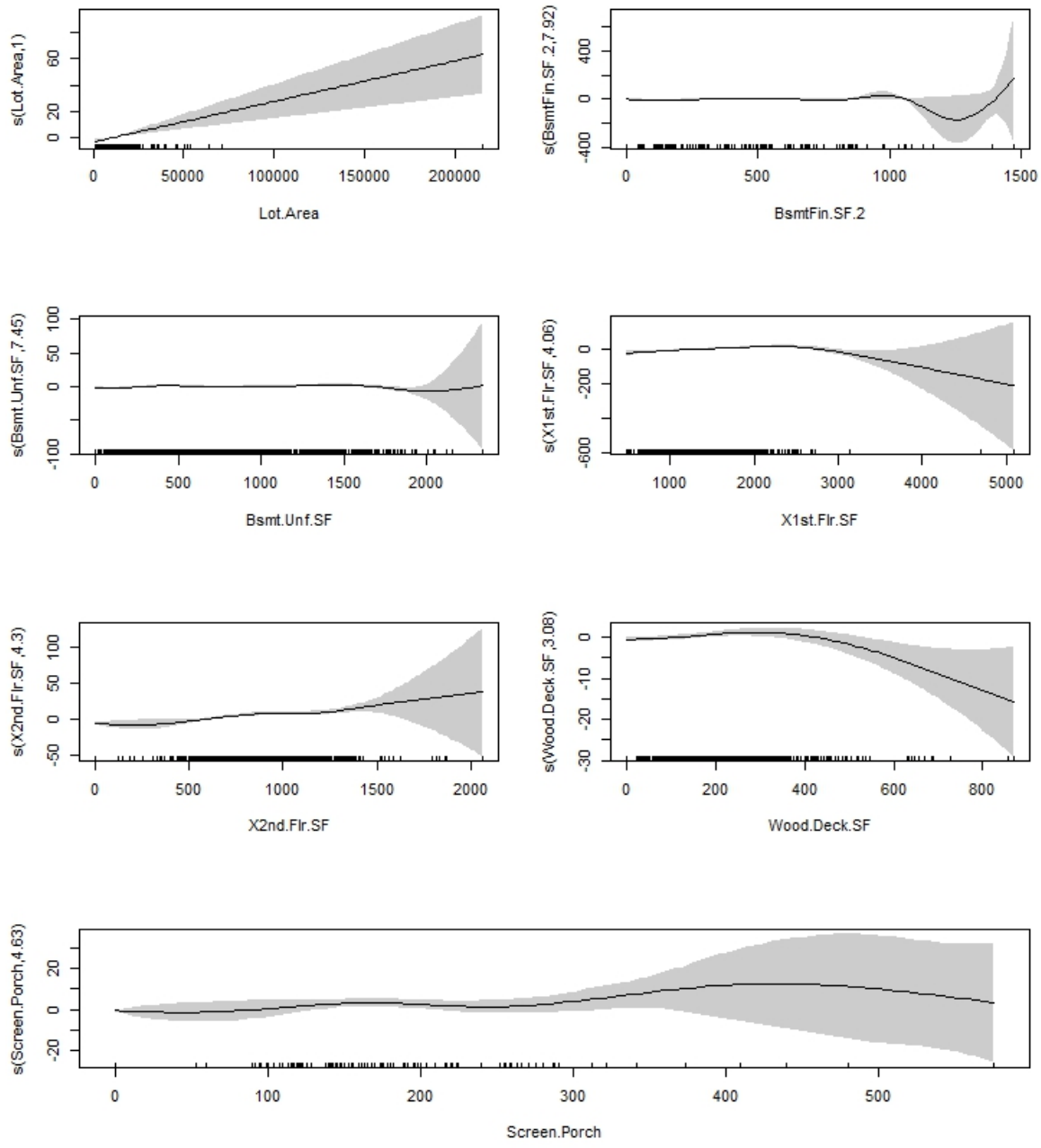

Gráfico 8: Funciones suavizadas de las variables continuas del modelo con los intervalos de confianza sombreados.

En el Gráfico 8, vemos que realmente la variable `Lot.Area` es una recta. En cambio, el resto, tienen curvatura. En el caso de esta variable recta, vemos que es creciente, es decir, a medida que se incrementan los pies cuadrados de la propiedad, se incrementa la probabilidad de que el precio de la vivienda esté por encima de la media. En cambio, en el caso de la variable `BsmtFin.SF.2` (pies cuadrados de la segunda valoración del terminado del sótano), vemos que

se mantiene casi constante muy cerca del cero hasta que, superando los 1000 pies cuadrados, el número de datos disponibles se reduce drasticamente y, en consecuencia, la curva estimada comienza a variar mucho. Vemos que a lo largo de casi toda la curva, el intervalo de confianza (zona sombreada), incluye al cero por lo que podemos decir que esta variable no tiene influencia real sobre el cambio de grupo de las viviendas. Esta misma situación se da en el caso de las varibles `Bsmt.Unf.SF` (pies cuadrados de sótano sin terminar) y `Screen.Porch` (pies cuadrados de porche semicerrado).

En el caso de las variables `X1st.Flr.SF` (pies cuadrados de la planta baja) y `Wood.Deck.SF` (pies cuadrados de porche de madera), ambas comienzan con valores por debajo del cero pero muy cercanos y ascienden ligeramente hasta que llegado un punto (2000 y 400 pies cuadrados, respectivamente), comienzan a decrecer. En ambos casos, el número de datos desde donde empiezan a decrecer las curvas es muy reducido, siendo más llamativo en el caso de los pies cuadrados de la planta baja. Esta falta de datos hace que los intervalos de confianza se amplien mucho.

Por último, vemos que la variable `X2nd.Flr.SF` (pies cuadrados de la primera planta) sufre un incremento reducido pero permanente alejándose del cero según se incrementan los pies cuadrados.

4.5. Comparación de modelos

En este apartado final, compararemos los modelos obtenidos en este capítulo. Para realizar la comparación, utilizaremos, en el caso de variable respuesta continua, el error de predicción y, en el caso de variable respuesta binaria, el error de clasificación.

Para calcular estos errores, en primer lugar, debemos dividir la muestra que tenemos en muestra de entrenamiento y muestra test. Para realizar eso, utilizaremos la siguiente secuencia:

```
> set.seed(1)
> train <- sample(c(TRUE,FALSE),nrow(datos),replace=TRUE)
> test <- !train
```

Siendo `train` la muestra de entrenamiento y `test` la muestra para testar el error. Así, `train` es una muestra de tamaño 604 y `test` de tamaño 555, conservando ambas el mismo número de variables que la muestra completa.

4.5.1. Variable respuesta continua

Calcularemos el error de predicción de la siguiente forma:

$$\text{error} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left(\frac{Y_i - \hat{Y}_i}{Y_i} \right)^2$$

siendo Y nuestra variable respuesta, $\hat{Y}$ la predicción de la Y, ambas contando solamente con los valores correspondientes a la muestra `test`, y N_{test} el tamaño de la muestra `test`. Decidimos relativizar el error ya que, en caso de no hacerlo, al elevar las diferencias al cuadrado y sumarlas, obtendríamos errores con muchas cifras, lo que dificultaría su comparación. Mientras que, al relativizarlo, van a resultar cifras pequeñas cercanas a cero.

4.5.1.1. Forward-stepwise

```
> testforrss <- numeric()
> for(i in 2:length(names(coef(bsscf,501)))){
+ testforrss[i] <- which(colnames(Resto)==names(coef(bsscf,501))[i])
+ }
> testforrss[1] <- 1

> testforcp <- numeric()
> for(i in 2:length(names(coef(bsscf,118)))){
+ testforcp[i] <- which(colnames(Resto)==names(coef(bsscf,118))[i])
+ }
> testforcp[1] <- 1

> testforbic <- numeric()
> for(i in 2:length(names(coef(bsscf,61)))){
+ testforbic[i] <- which(colnames(Resto)==names(coef(bsscf,61))[i])
+ }
> testforbic[1] <- 1

> testforr2 <- numeric()
> for(i in 2:length(names(coef(bsscf,215)))){
+ testforr2[i] <- which(colnames(Resto)==names(coef(bsscf,215))[i])
+ }
> testforr2[1] <- 1

> errorf2rss <- mean(((SalePrice[test] - Resto[test,testforrss]%*%t(t(coef(bsscf,501))))/
+ SalePrice[test])^2)

[1] 26.75648

> errorf2cp <- mean(((SalePrice[test] - Resto[test,testforcp]%*%t(t(coef(bsscf,118))))/
+ SalePrice[test])^2)
```

```
[1] 0.09390

> errorf2bic <- mean(((SalePrice[test] - Resto[test,testforbic]%*%t(t(coef(bsscf,61))))/
+ SalePrice[test])^2)

[1] 1.58343

> errorf2r2 <- mean(((SalePrice[test] - Resto[test,testforr2]%*%t(t(coef(bsscf,215))))/
+ SalePrice[test])^2)

[1] 0.05748
```

4.5.1.2. Backward-stepwise

```
> testbackrss <- numeric()
> for(i in 2:length(names(coef(bsscb,501)))){
+ testbackrss[i] <- which(colnames(Resto)==names(coef(bsscb,501))[i])
+ }
> testbackrss[1] <- 1

> testbackcp <- numeric()
> for(i in 2:length(names(coef(bsscb,121)))){
+ testbackcp[i] <- which(colnames(Resto)==names(coef(bsscb,121))[i])
+ }
> testbackcp[1] <- 1

> testbackbic <- numeric()
> for(i in 2:length(names(coef(bsscb,57)))){
+ testbackbic[i] <- which(colnames(Resto)==names(coef(bsscb,57))[i])
+ }
> testbackbic[1] <- 1

> testbackr2 <- numeric()
> for(i in 2:length(names(coef(bsscb,268)))){
+ testbackr2[i] <- which(colnames(Resto)==names(coef(bsscb,268))[i])
+ }
> testbackr2[1] <- 1

> errorb2rss <- mean(((SalePrice[test] - Resto[test,testbackrss]%*%t(t(coef(bsscb,501))))/
+ SalePrice[test])^2)

[1] 24.60922

> errorb2cp <- mean(((SalePrice[test] - Resto[test,testbackcp]%*%t(t(coef(bsscb,121))))/
+ SalePrice[test])^2)

[1] 1.39999
```

```
> errorb2bic <- mean(((SalePrice[test] - Resto[test,testbackbic]%*%t(t(coef(bsscb,57))))/
+ SalePrice[test])^2)

[1] 1.82637

> errorb2r2 <- mean(((SalePrice[test] - Resto[test,testbackr2]%*%t(t(coef(bsscb,268))))/
+ SalePrice[test])^2)

[1] 0.30396
```

4.5.1.3. Regresión Ridge

```
> rrc.pred <- predict(rrc,Resto[test,],s=lambda.rrc,type="response")
> rrc.pred.1se <- predict(rrc,Resto[test,],s=lambda.rrc.1se,type="response")

> error.rr <- mean(((SalePrice[test]-rrc.pred)/SalePrice[test])^2)

[1] 0.06763

> error.rr.1se <- mean(((SalePrice[test]-rrc.pred.1se)/SalePrice[test])^2)

[1] 0.12922
```

4.5.1.4. Lasso

```
> lassoc.pred <- predict(lassoc,Resto[test,],s=lambda.lassoc,type="response",alpha=1)
> lassoc.pred.1se <- predict(lassoc,Resto[test,],s=lambda.lassoc.1se,type="response",
+                            alpha=1)

> error.lasso <- mean(((SalePrice[test]-lassoc.pred)/SalePrice[test])^2)

[1] 0.08469

> error.lasso.1se <- mean(((SalePrice[test]-lassoc.pred.1se)/SalePrice[test])^2)

[1] 0.19047
```

4.5.1.5. LAR

```
> pred.larc.cp <- predict(larc,Resto[test,],s=which.min(larc$Cp),type="fit")
> pred.larc.RSS <- predict(larc,Resto[test,],s=which.min(larc$RSS),type="fit")

> error.lar.cp <- mean(((SalePrice[test]-pred.larc.cp$fit)/SalePrice[test])^2)

[1] 0.03640

> error.lar.RSS <- mean(((SalePrice[test]-pred.larc.RSS$fit)/SalePrice[test])^2)

[1] 0.00658
```

4.5.1.6. Componentes principales

```
> pred.pcacv <- predict(pcreg,ncomp=142,newdata=Resto[test,colSums(Resto !=0) > 2],type="response")
> pred.pca90 <- predict(pcreg,ncomp=213,newdata=Resto[test,colSums(Resto !=0) > 2],type="response")

> error.pcacv <- mean(((SalePrice[test]-pred.pcacv)/SalePrice[test])^2)

[1] 0.08138

> error.pca90 <- mean(((SalePrice[test]-pred.pca90)/SalePrice[test])^2)

[1] 0.05643
```

4.5.1.7. Mínimos Cuadrados Parciales

```
> pred.plscv <- predict(plsreg,ncomp=2,newdata=Resto[test,colSums(Resto !=0) > 2],type="response")
> pred.pls90 <- predict(plsreg,ncomp=282,newdata=Resto[test,colSums(Resto !=0) > 2],type="response")

> error.plscv <- mean(((SalePrice[test]-pred.plscv)/SalePrice[test])^2)

[1] 0.08996


> error.pls90 <- mean(((SalePrice[test]-pred.pls90)/SalePrice[test])^2)

[1] 0.01189
```

4.5.1.8. GAM

```
> pred.gamc <- predict(gamc.final,Covar[test,],type="response")

> error.gamc <- mean(((SalePrice[test]-pred.gamc)/SalePrice[test])^2)

[1] 0.00812
```

4.5.1.9. Conclusiones

Podemos ver en la Tabla 3, que el método Forward-stepwise tiene un error de predicción muy alto cuando el número de variables (`M`) que introducimos en el modelo es bajo o muy alto. En cambio, cuando es un número intermedio de covariables, en relación al número total, el error es reducido, siendo, por ejemplo, en el caso de 215 variables, de 0.05748. Sucede algo parecido con el método Backward-stepwise, pero en este caso los errores para los números intermedios de variables son mucho más elevados.

En el caso de la regresión Ridge, con el λ `min` obtenemos un error de predicción bajo. En cambio, con λ `1 se`, el error ya comienza a ser considerable. Con el método lasso sucede lo mismo, tiene un error de predicción reducido y otro demasiado alto. En cambio, con el métod Least Angle Regression, obtenemos los errores más bajos, siendo el que conseguimos con el λ que minimiza la RSS de 0.00658.

Con el método de componentes principales obtenemos unos buenos errores de predicción, aunque no son los más bajos de entre todos los modelos, igual que sucede con los obtenidos por el método de mínimos cuadrados parciales. Pero cabe destacar, que en este último método, con solamente dos variables obtenemos un error muy reducido, a diferencia de lo que ocurría en los casos anteriores, que cuando tenían un número muy alto o muy bajo de variables el error de predicción se disparaba. Por último, el modelo GAM da un error muy reducido, estando ligeramente por encima del que obteníamos con el modelo LAR.

A pesar de que con el método Forward-stepwise obtenemos algunos buenos resultados según el número de variables que introduzcamos en el modelo, no es un método recomendable ya que, como hemos explicado anteriormente, no asegura obtener el subconjunto de variables óptimo, ya que una vez que se incorpora una variable al modelo, no se puede descartar. Vemos que con los métodos de regularización y de reducción de la dimensión, obtenemos unos buenos resultados, pero solamente el método lasso consigue reducir el número de covariables a la vez que facilita la interpretación de los resultados, ya que, por ejemplo, el método de componentes principales reduce considerablemente el número de variables, pero su interpretación es bastante compleja. En el caso del método GAM, a pesar de que para reducir el número de covariables hay que combinarlo con algún método de selección de subconjuntos, el resultado que obtenemos

en términos de error de predicción es muy bueno, pero este tiene el problema de que el tiempo de computación es ligeramente elevado.

En conclusión, observando los resultados y las características de cada método, nos quedaríamos con el método lasso o, en un segundo lugar, con el modelo GAM.

En cuanto a las variables que incluímos en cada modelo, hay algunas que son comunes a todos, exceptuando para los métodos Forward- y Backward-stepwise. Estas son: `Mas.Vnr.Area` (pies cuadrados de los distintos tipos de acabo de la fachada), `Exter.QualTA` (la calidad de los materiales del exterior está en la media), `Total.Bsmt.SF` (pies cuadrados del sótano), `X1st.Flr.SF` (pies cuadrados de la planta baja de la vivienda), `Gr.Liv.Area` (pies cuadrados de área habitable por encima del nivel del suelo), `TotRms.AbvGrd15` (que haya 15 habitaciones por encima del nivel del suelo sin contar los baños) y `Garage.Cars3` (en el garage se pueden aparcar tres coches).

Método		**Parámetro**	**Error de predicción**
Forward	**BIC**	M = 61	1.58343
	C_p	M = 118	0.09390
	R^2 ajustado	M = 215	0.05748
	RSS	M = 501	26.75648
Backward	**BIC**	M = 57	1.82637
	C_p	M = 121	1.39999
	R^2 ajustado	M = 268	0.30396
	RSS	M = 501	24.60922
Ridge	**λ min**	λ = 93185.89	0.06763
	λ 1 se	λ = 342773.1	0.12922
Lasso	**λ min**	λ = 1518.698	0.08469
	λ 1 se	λ = 10714.11	0.19047
LAR	**C_p**	λ = 315.8383	0.03641
	RSS	λ = 474329765543	0.00658
PCA	**CV**	M = 142	0.08138
	90 %	M = 213	0.05643
PLS	**CV**	M = 2	0.08996405
	90 %	M = 282	0.01189389
GAM			0.00812

Tabla 3: Comparación de los errores de predicción de los distintos modelos.

4.5.2. Variable respuesta binaria

En el caso de variable respuesta binaria, utilizaremos el error de clasificación (CE), definido de la siguiente forma:

$$\begin{aligned} CE &= 1 \quad \text{si } Y = 1 \text{ y } \hat{Y} < \frac{1}{2} \text{ o } Y = 0 \text{ y } \hat{Y} > \frac{1}{2}, \\ &= \frac{1}{2} \quad \text{si } \hat{Y} = \frac{1}{2}, \\ &= 0 \quad \text{en otro caso.} \end{aligned}$$

Hemos implementado estos cálculos en la siguiente función:

```
pred <- function(y,newdata){
  CE <- numeric()
  for(i in 1:length(y)){
    if(newdata[i]==1/2){CE[i] <- 1/2}
    else if(y[i]==1 & newdata[i]>1/2){CE[i] <- 0}
    else if(y[i]==0 & newdata[i]<1/2){CE[i] <- 0}
    else{CE[i] <- 1}
  }
  print(mean(CE))
}
```

4.5.2.1. Regresión Ridge

```
> rr.pred <- predict(rr,Resto[test,],s=lambda.rr,type="response")
> rr.pred.1se <- predict(rr,Resto[test,],s=lambda.rr.1se,type="response")
> error.rr <- pred(Price[test],rr.pred)
[1] 0.04144
> error.rr.1se <- pred(Price[test],rr.pred.1se)
[1] 0.05586
```

4.5.2.2. Lasso

```
> lasso.pred <- predict(lasso,Resto[test,],s=lambda.lasso,type="response",alpha=1)
> lasso.pred.1se <- predict(lasso,Resto[test,],s=lambda.lasso.1se,type="response",alpha=1)
> error.lasso <- pred(Price[test],lasso.pred)
[1] 0.10090
> error.lasso.1se <- pred(Price[test],lasso.pred.1se)
[1] 0.10991
```

4.5.2.3. LAR

```
> yest.df <- lar$beta[1,92] + lar$X %*% lar$beta[,92][-1]
> yest.gdf <- lar$beta[1,92] + lar$X %*% lar$beta[,210][-1]

> error.lar.df <- pred(Price[test],yest.df)

[1] 0.43964

> error.lar.gdf <- pred(Price[test],yest.gdf)

[1] 0.44685
```

4.5.2.4. PLS

```
> yest.rss <- pls$tt[,1:20]%*%pls$CoeffC[1:20]
> yest.aic <- pls$tt[,1:11]%*%pls$CoeffC[1:11]

> error.pls.rss <- pred(Price[test],yest.rss)

[1] 0.46667

> error.pls.aic <- pred(Price[test],yest.aic)

[1] 0.46306
```

4.5.2.5. GAM

```
> pred.gam <- predict(gam.final,Covar[test,],type="response")

> error.pred.gam <- pred(Price[test],pred.gam)

[1] 0.03784
```

Método		**Parámetro**	**Error de clasificación**
Ridge	λ **min**	$\lambda = 0.05690$	0.04144
	λ **1 se**	$\lambda = 0.22972$	0.05586
Lasso	λ **min**	$\lambda = 0.03492$	0.10090
	λ **1 se**	$\lambda = 0.04616$	0.10991
LAR	**df**	$\gamma = 3.217$	0.43964
	gdf	$\gamma = 1.224$	0.44685
PLS	**RSS**	5.265e-12	0.46486
	AIC	22.00009	0.46847
GAM			0.03784

Tabla 4: Comparación de los errores de clasificación de los distintos modelos.

4.5.2.6. Conclusiones

Podemos ver en la tabla 4, que los métodos con los que obtenemos un menor error de clasificación son la regresión Ridge y la GAM, siendo este segundo modelo el que obtiene el mejor resultado. Vemos, en cambio, que los errores de predicción del resto de métodos son muy elevados, incluso en el caso del método Lasso, casi alcanzando, en el caso de los métodos LAR y PLS, el 0.5.

En este caso, no hay ninguna variable que mantengamos en todos los modelos. En cambio, si no tenemos en cuenta el modelo GAM, el resto de modelos sí comparten algunas, que son: `Overall.Qual8` (todos los materiales y el nivel de terminado de la casa son muy buenos), `Overall.Qual9` (todos los materiales y el nivel de terminado de la casa son excelentes), `Exter.QualTA` (la calidad de los materiales del exterior está en la media), `Kitchen.QualTA` (la calidad de la cocina está en la media) y `Garage.Cars3` (en el garage se pueden aparcar tres coches). A su vez, podemos ver que las variables `Exter.QualTA` y `Garage.Cars3` se encuentra en los modelos obtenidos tanto con variable respuesta continua como binaria.

Bibliografía

[1] Akaike, H. (1973), *Information theory and an extension of the maximum likelihood principle*, in Petrov, B.N. and Csáki, F., editors, 2nd International Symposium on Information Theory, Budapest: Akadémiai Kiadó, p. 267-281.

[2] 'Ames, IA Real Estate Data' submitted by Dean De Cock, Department of Mathematics and Computer Science, Truman State University. Dataset obtained from the Journal of Statistics Education (http://www.amstat.org/publications/jse).

[3] Augugliaro, L. (2014), *Differential Geometric LARS (dgLARS) method*, R package version 1.0.5.

[4] de Boor, C. (1978), *A Practical Guide to Splines*, Springer, Berlin.

[5] Cadarso-Suárez, C. (2015), Notas de la asignatura Estadística no paramétrica (Máster en Técnicas Estadísticas e Investigación Operativa).

[6] Chen, S. S., Donoho, D. L. and Saunders, M. A. (1998), Atomic decomposition by basis pursuit, *SIAM Journal on Scientific Computing*, 33-61.

[7] Chouldechova, A. and Hastie, T. (2015), Generalized Additive Model Selection, *The Annals of Applied Statistics*.

[8] Craven, P. and Wahba, G. (1979), Smoothing noisy data with spline functions, *Numer. Math.*, 31, 377-403.

[9] Crujeiras, R.M. and Sánchez, C.A. (2015), Notas de la asignatura Modelos de Regresión (Máster en Técnicas Estadísticas e Investigación Operativa).

[10] deLeeuw, J. 1992. Introduction to Akaike (1973) Information Theory and an Extension of the Maximum Likelihood Principle. Pages 599-609 In: Kotz, S., and N.L. Johnson, editors. *Breakthroughs in Statistics Volume 1. Foundations and Basic Theory*. Springer Series in Statistics, Perspectives in Statistics. Springer-Verlag: New York.

[11] Durbán, M. (2008), *Splines con Penalizaciones (P-splines)*, Ediciones Universidad Pública de Navarra (Ed. Dolores Ugarte).

[12] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004), Least angle regression, *Annals of Statistics*, 32(2), 407-499.

[13] Eilers, P.H.C. and Marx, B. D., (1996), Flexible smoothing using B-splines and penalties, *Statistical Science*, 11, 89-121.

[14] Frank, I. and Friedman, J. (1993), A statistical view of some chemometrics regression tools (with discussion), *Technometrics*, 35(2), 109-148.

[15] Friedman, I., Hastie, T., Simon, N. and Tibshirani, R. (2016), *Lasso and Elastic-Net Regularized Generalized Linear Models*, R package version 2.0-5.

[16] Friedman, I., Hastie, T. and Tibshirani, R. (2008), *The elements of statistical learning*, Springer.

[17] Friedman, I., Hastie, T. and Tibshirani, R. (2010), Regularization Paths for Generalized Linear Models via Coordinate Descent *Journal of Statistical Software*, 33,1.

[18] González-Manteiga, W. (2015), Notas de la asignatura Análisis Multivariante (Máster en Técnicas Estadísticas e Investigación Operativa).

[19] Hatie, T. and Efron, B. (2013), *Least Angle Regression, Lasso and Forward Stagewise*, R package version 1.2.

[20] Hastie T., Tibshirani R. (1990), *Generalized Additive Models*, Chapman-Hall.

[21] Hoerl, A. E. and Kennard, R. (1970), Ridge regression: biased estimation for nonorthogonal problems, *Technometrics*, 12, 55-67.

[22] Izenman, A. (1975), Reduced-rank regression for the multivariate linear model, *Journal of Multivariate Analysis*, 5, 248-264.

[23] James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013), *An introduction to statistical learning*, Springer.

[24] Lang, S., Brezger, A. (2004), Bayesian P-Splines, *Journal of Computational and Graphical Statistics*, 13, 183-212.

[25] le Cessie, S. and van Houwelingen, J.C. (1990), Ridge Estimators in Logistic Regression, *Journal of Applied Statistics*, 41(1), 191-201.

[26] Lin, Y. and Zhang, H. H. (2003), *Component selection and smoothing in smoothing spline analysis of variance models, Technical report*, Department of Statistics, University of Wisconsin, Madison.

[27] Mallows, C. (1973), Some comments on Cp, *Technometrics*, 15(4), 661-675.

[28] Meier, L., Van de Geer, S. and Bühlmann, P. (2009), High-dimensional additive modeling, *The Annals of Statistics*, 37(6B), 3779-3821.

[29] Mevik, B-H., Wehrens, R and Hovde Liland, K. (2015), *Partial Least Squares and Principal Component Regression*, R package version 2.5-0.

[30] Meyer, N., Maumy-Bertrand, M. and Bertrand, F. (2010), Comparaison de variantes de régressions logistiques PLS et de régression PLS sur variables qualitatives: application aux données d?allélotypage, *Journal de la Société Française de Statistique*, 151(2).

[31] Moulton, L.H., Weissfeld, L.A. and St.Laurent, R.T. (1993), Bartlett correction factors in logistic regression models, *Computational Statistics and Data Analysis*, 15, 01-11.

[32] Ravikumar, P., Liu, H., Lafferty, J. and Wasserman, L. (2008), *Sparse additive models*, Technical report, Carnegie Mellon University.

[33] Sarkar, S.K., Midi, H. and Sohel, R. (2010), Model Selection in Logistic Regression and Performance of its Predictive Ability, *Australian Journal of Basis and Applied Sciences*, 4(12), 5813-5822.

[34] Schwarz, G. (1978), Estimating the dimension of a model, *The Annals of Statistics*, 6(2), 461-464.

[35] Thomas Lumley using Fortran code by Alan Miller (2009), *leaps: regression subset selection*, R package version 2.9.

[36] Tibshirani, R. (1996), Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society*, Series B, 58, 267-288.

[37] Yee, T. and Wild, C. (1996), Vector generalized additive models, *Journal of the Royal Statistical Society*, Series B, 58, 481-493.

[38] Wold, S., Sjöström, M. and Eriksson, L. (2001), PLS-regression: a basic tool of chemometrics, *Chemometrics and Intelligent Laboratory Systems*, Series B, 58, 109-130.

[39] Wood, S.N. (2006), *Generalized Additive Models. An introduction with R*, CRC/Chapman- Hall.

[40] Wood, S.N. (2016), *mgcv: Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation*, R package version 1.8-13.

[41] Zou, H. and Hastie, T. (2005), Regularization and variable selection via the elastic net, *Journal of the Royal Statistical Society*, Series B, 67(2), 301-320.

[42] Zou, H., Hastie, T. and Tibshirani, R. (2007), On the degrees of freedom of the lasso, *The Annals of Statistics*, 35(5), 2173-2192.