

Índice

1. Introducción a la Inteligencia Artificial: Técnicas Machine Learning	11
1.1. Introducción	11
1.2. Árboles de Clasificación y Regresión (<i>Classification and Regression Trees, CARTs</i>)	13
1.2.1. Construcción de un Árbol	13
1.2.2. Árboles óptimos	21
1.3. Bosques Aleatorios (<i>Random Forests, RFs</i>)	25
1.4. Árboles de Regresión Lógica	28
1.5. Regresión a Trazos Adaptativa Multivariante (<i>Multivariate Adaptive Regression Splines, MARS</i>)	30
1.6. Lecturas Recomendadas	33
2. k-vecinos más cercanos: Clasificación y Predicción	35
2.1. Introducción	35
2.2. Matriz de Confusión	36
2.3. Método k vecinos más cercanos	37
2.4. Lecturas Recomendadas	38
3. Regresión Lineal con técnicas Machine Learning	39
3.1. Introducción	39
3.2. Regresión Lineal con Prueba y Error	40
3.3. Tamaño muestral muy elevado	42
3.4. Sobre-ajuste	43
3.5. Lecturas Recomendadas	45
4. Selección de Predictores: Regresión Lasso, Ridge y Elastic Net	47
4.1. Introducción	47
4.2. Regresión Ridge	48
4.3. Regresión Lasso	50

4.4. Elastic Net	51
4.5. Lecturas Recomendadas	51
5. Regresión Logística con técnicas Machine Learning	53
5.1. Introducción	53
5.2. Lecturas Recomendadas	56
6. Deep Learning: Redes Neuronales Artificiales	57
6.1. Introducción	57
6.1.1. Formulación matemática de una Red Neuronal	58
6.2. Redes Neuronales con R: librerías nnet y neuralnet	59
6.3. Redes neuronales e inteligencia artificial	61
6.4. Operador tubería	63
6.5. Redes Neuronales Convolucionales	63
6.5.1. Redes Neuronales Convolucionales con R	66
6.6. Lecturas Recomendadas	73

Capítulo 1

Introducción a la Inteligencia Artificial: Técnicas Machine Learning

1.1. Introducción

Comenzaremos el estudio de las técnicas de Inteligencia Artificial, a veces denominada simplemente con su acrónimo IA, estudiando en este capítulo *Machine Learning*, es decir, *Aprendizaje Automático* o *Maestro en determinar Patrones*, que es una disciplina del campo de la Inteligencia Artificial que dota a los ordenadores de la capacidad de identificar patrones en datos masivos y elaborar predicciones (análisis predictivo) mediante el uso de modelos a través de algoritmos.

Estos datos, utilizados con este propósito de la determinación de patrones, se denominan *Datos de Aprendizaje o Entrenamiento*. A Machine Learning se la denomina también *Statistical Learning*; esta denominación procede del campo de la Estadística mientras que *Machine Learning* procede de la informática.

Es interesante destacar que el ordenador es el que crea los algoritmos basados en los datos de entrenamiento que se utilizan para calibrar el modelo de aprendizaje automático. Con base los datos de entrenamiento, el algoritmo encuentra las reglas subyacentes internamente.

Este aprendizaje permite a los ordenadores realizar tareas específicas de forma autónoma, es decir, sin necesidad de ser programados.

Cuando los datos de aprendizaje no han sido estructurados ni etiquetados por nosotros, este método de creación del modelo se denomina *Deep Learning* (*Aprendizaje Profundo*). Este tipo de aprendizaje automático se realiza con el uso de las denominadas *Redes Neuronales Artificiales*, *Neural Networks*.

El término se utilizó por primera vez en 1959, cuando los ordenadores eran muy limitados. Dada a la gran capacidad cálculo de los ordenadores en la actualidad y a la gran cantidad de datos que puede manejar, esta disciplina ha ganado relevancia en los últimos tiempos. Las técnicas de Aprendizaje Automático son, de hecho, una parte fundamental del denominado *Big Data Analysis*.

Los algoritmos de Machine Learning, en ocasiones abreviado como ML, se dividen en dos categorías:

Aprendizaje supervisado: Tiene como propósito predecir una variable observada a partir de otras variables relacionadas. Estos algoritmos cuentan con un aprendizaje previo basado en un sistema de etiquetas asociadas a unos datos que les permiten tomar decisiones o hacer predicciones. Un ejemplo es un detector de spam que etiqueta un e-mail como spam o no, dependiendo de los patrones que ha aprendido del histórico de correos (remitente, relación texto/imágenes, palabras clave en el asunto, etc.).

Los algoritmos de Aprendizaje Supervisado se dividen en Algoritmos de Regresión (cuando la variable predicha es numérica) y Algoritmos de Clasificación (cuando la variable predicha es categórica). Son Algoritmos de Regresión, la Regresión Lineal y el Random Forest.

Son Algoritmos de Clasificación la Regresión Logística y los Árboles de Decisión.

Aprendizaje no supervisado: Estos algoritmos no cuentan con un conocimiento previo. Se enfrentan al caos de datos con el objetivo de encontrar patrones que permitan organizarlos de alguna manera. Por ejemplo, en el campo del marketing se utilizan para extraer patrones de datos masivos provenientes de las redes sociales y crear campañas de publicidad altamente segmentadas.

Dada la reciente implantación de las técnicas estudiadas en este libro, en muchas ocasiones hemos mantenido la denominación anglosajona para su fácil identificación. También queremos decir que los Métodos estudiados en este libro suelen venir englobados habitualmente en lo que se denomina *Minería de Datos* o *Data Mining* porque suelen utilizarse con datos de gran dimensión (número p de variables muy alto) y/o enorme tamaño muestral (n muy grande) y, en ocasiones, con $p \gg n$ lo que crea graves problemas de aplicación de las técnicas estadísticas tradicionales.

Una cuestión de interés que nos gustaría resaltar es que, en contra de lo que comúnmente se cree, este tipo de técnicas suelen ser poco robustas, es decir, suelen ser sensibles a la presencia de datos anómalos en la muestra.

También nos gustaría mencionar la referencia habitual de este tipo de técnicas, Hastie et al. (2009), aunque los objetivos de este libro y el que está leyendo son distintos.

1.2. Árboles de Clasificación y Regresión (*Classification and Regression Trees, CARTs*)

Los Árboles de Clasificación y Regresión (*Classification and Regression Trees*), habitualmente conocidos por su acrónimo anglosajón CARTs, son una técnica consistente en descubrir relaciones (condicionales) entre un gran número de covariables explicativas X_i y una dependiente Y , cualitativa o continua. Se denominan *Árboles de Clasificación* cuando se aplican a variables dependientes cualitativas y *Árboles de Regresión* cuando la variable dependiente es de tipo continuo.

Ambas técnicas, debidas a Breiman et al. (1993), suponen la aplicación de un algoritmo que va dividiendo el conjunto de individuos de la muestra en subgrupos, de manera que se minimice la heterogeneidad (denominada *impureza del nodo*) dentro de los nuevos grupos formados.

Esta técnica ordena por importancia a las covariables X_i , lo que implica que podemos quedarnos con sólo unas pocas covariables explicativas antes de hacer una regresión de modelos GLM o GAM. (Para un estudio de estos modelos, se recomienda el texto AEQR.)

1.2.1. Construcción de un Árbol

Los datos de los que dispondremos serán n observaciones de una variable dependiente (que, en estudios de Salud, será un fenotipo) Y y p variables independientes $X_1, \dots, X_p$ las cuales pensamos sirven para predecir a la variable dependiente.

Un Árbol se construye determinando primero la variable X_j *más predictiva* del Y en el sentido que veremos más abajo. Supongamos para comenzar que esta variable X_j tomara sólo dos valores; los individuos de la muestra, los cuales inicialmente están todos en un conjunto, denominado *nodo raíz*, o *nodo padre* y que representaremos por Ω , se dividirán en dos subconjuntos o *nodos hija*, Ω_1 y Ω_2 según los valores de esta variable de X_j . Si tomase más valores –por ejemplo, fuera de tipo continua–, los dos grupos se formarían dependiendo de si $X_j < c$, ó $X_j \geq c$, siendo c algún valor posible de X_j . Se elige a continuación la segunda variable más predictiva de Y en cada uno de los nodos hija y se aplica de nuevo una regla similar en cada uno de los dos nodos hija; y así se sigue particionando la muestra hasta un determinado momento fijado por una *regla de parada* (por ejemplo que el nodo tenga menos de tres individuos). Advertimos que el método puede conducir a árboles asimétricos.

Cuando hayamos construido el Árbol habremos seleccionado unas cuantas covariables, las más influyentes en la variable dependiente, y además por orden

de importancia.

La elección de la variable más predictiva y la regla de clasificación a partir de ella, se basa en lo que se denomina *impureza del nodo* (es decir, su heterogeneidad) $I(\Omega)$ para la que existen varias opciones; no obstante, suele decirse que todas conducen básicamente al mismo árbol. La variable *más predictiva* (y la regla de clasificación basada en ella) se elige como aquella que maximice

$$I(\Omega) - I(\Omega_D) - I(\Omega_I)$$

siendo Ω_D y Ω_I los dos nodos hija (Derecha e Izquierda) del nodo Ω obtenidos después de aplicar la regla considerada.

Variable dependiente dicotómica

De las diversas medidas de impureza del nodo, la más habitual para el caso de que la variable dependiente Y sea dicotómica, es el *Índice de Gini* definido por

$$I(\Omega) = 2 p_{\Omega} (1 - p_{\Omega})$$

en donde $p_{\Omega} = P(Y = 1|\Omega)$ es la probabilidad de que el fenotipo Y sea 1, condicionado a pertenecer al nodo Ω .

Ejemplo 1.1

Como ilustración de estos conceptos supongamos 100 individuos en los que se observa una variable dependiente dicotómica Y , como por ejemplo estar Enfermo-No enfermo, y $p = 3$ variables independientes dicotómicas X_1, X_2, X_3 , tipo Presencia-Ausencia.

Nuestra matriz de datos será de la forma

Y	X_1	X_2	X_3
1	0	1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	1	0	1
100			

datos que, expresados en forma de tabla, sería de la forma por ejemplo,

	$X_1 = 1, X_1 = 0$ (Joven-Viejo)		$X_2 = 1, X_2 = 0$ (Mutación en un gen)		$X_3 = 1, X_3 = 0$ (Sexo)		
$Y = 1$	55	35	65	15	35	35	
$Y = 0$	5	5	20	0	15	15	
	60	40	85	15	50	50	100

en donde hemos dado nombre a las variables dicotómicas predictoras. Si, por las razones que más abajo veremos, la variable más predictiva en el grupo A, fuera X_2 , los 100 individuos que están en el nodo madre A, se dividirían en dos grupos según los dos valores de la variable

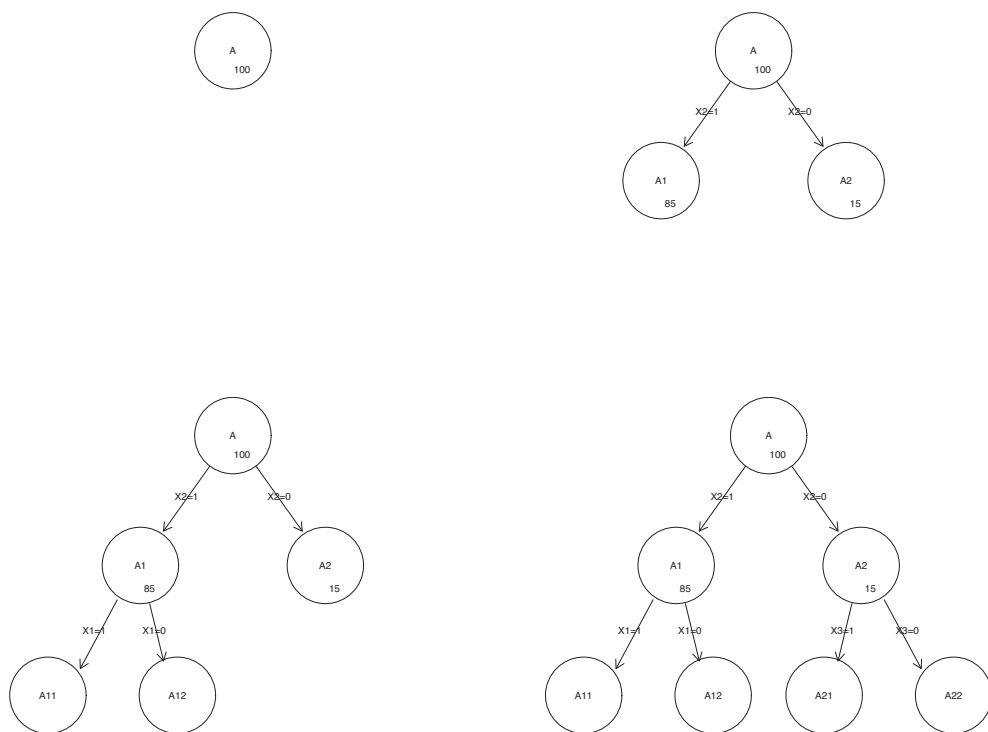


Figura 1.1 : Proceso de construcción de un Árbol

X_2 , es decir, en dos grupos de 85 y 15 individuos respectivamente, como aparece en el gráfico superior derecha de la Figura 1.1.

Después analizaríamos cuál, de las tres variables, es la más predictiva en el grupo A1. Si en este grupo la variable más predictiva fuera X_1 , este grupo se dividiría en dos grupos, A11 y A12 según los dos valores de X_1 , como aparece en el gráfico de abajo a la izquierda de la Figura 1.1.

Por último se analiza cuál es la variable más predictiva en el grupo A2 y, si ésta fuera X_3 (que podría ser cualquiera de las tres incluso una de las ya utilizadas anteriormente), el grupo A2 se dividiría en dos grupos según los dos valores de X_3 , como aparece en el gráfico de abajo a la derecha de la Figura 1.1. Y así seguiríamos hasta alcanzar alguna regla de parada fijada previamente.

Como dijimos antes, el encontrar la variable X que maximice la Impureza de un Nodo (más discrimine) es medida por

$$i(A) - 0'5 \cdot i(A1) - 0'5 \cdot i(A2)$$

siendo $i(B)$ el Índice de Gini de un nodo:

$$i(A) = 2p(1-p) \approx 2\hat{p}(1-\hat{p})$$

Con estas opciones obtendríamos el valor de la *Importancia Global Estandarizada de cada Covariable* $\hat{\theta}_j/SE(\hat{\theta}_j)$. Su representación gráfica ordenada la obtenemos con la función `varImpPlot`.

Es de destacar que, como esta función realiza, de hecho, un remuestreo bootstrap, los resultados obtenidos en cada ocasión pueden ser distintos y, a su vez, distintos de los que obtenemos aquí.

Ejemplo 1.6

Consideremos de nuevo los datos Virco y la variable dependiente NFV.Fold-IDV.Fold como anteriormente. La función `randomForest` no admite datos perdidos por lo que cambiamos un poco las sentencias de dicho ejemplo y modificamos los datos considerados. Ejecutando (1) obtenemos la importancia de cada variable que vemos representada en la Figura 1.8 ejecutando (2).

```
> Rasgo22<-Rasgo2[!is.na(Rasgo2)]
> Vircogen22<-Vircogen[!is.na(Rasgo2),]

> Bosque1<-randomForest(Vircogen22,Rasgo22,importance=T) (1)
```

Call:

```
randomForest(x = Vircogen22, y = Rasgo22, importance = T)
      Type of random forest: regression
      Number of trees: 500
```

No. of variables tried at each split: 33

```
      Mean of squared residuals: 5688.114
      % Var explained: 13.77
```

```
> varImpPlot(Bosque1) (2)
```

En el gráfico de la izquierda de esta figura se ve que las variables más importantes (en esta selección aleatoria bootstrap) han resultado ser P35, P36, P54, P73 y P20.

Aunque la función `randomForest` no admite datos perdidos, hay algunas posibilidades de actuación si los hubiera.

1.4. Árboles de Regresión Lógica

Este tipo de análisis, en búsqueda de las covariables más predictivas de tipo dicotómico, se debe a Ruczinski y otros (2003, 2004) y se basa en lo que se denomina *Combinaciones Booleanas*. Si por ejemplo $X_1, \dots, X_4$ son 4 variables dicotómicas, o mejor, las variables indicadoras de 4 variables dicotómicas, una Combinación Booleana de ellas podría ser

$$(X_1 \wedge X_2) \vee (X_3^c \wedge X_4) \quad [1.1]$$

en donde los operadores lógicos que se utilizan son $\vee = \text{o}$, $\wedge = \text{y}$ y $A^c = \text{no } A$ y que se leería diciendo que, o bien es $X_1 = 1$ y $X_2 = 1$, o bien es $X_3 = 0$ y $X_4 = 1$. Esta dicotomía recurrente se expresa en forma de árbol de decisión (como la Figura 1.9 del siguiente ejemplo), consistiendo el método de los *Árboles de Regresión Lógica* en encontrar la Combinación Booleana más predictiva; es decir, como si dijéramos el “modelo lineal generalizado” más predictivo para variables dicotómicas, de la forma

$$g(E[Y]) = \beta_0 + \sum_{j=1}^t \beta_j L_j \quad [1.2]$$

siendo los L_j Combinaciones Booleanas de las variables predictivas de tipo indicador como las de [1.1].

Este método se puede ejecutar con la función `logreg` de la librería `LogicReg`

`logreg(resp,bin,select)`

en donde los argumentos principales de esta función son **resp**, para indicar el nombre de la variable dependiente; **bin**, para los datos en formato data frame, y **select**, para indicar el tipo de modelo a ajustar, es decir, el número de árboles de regresión lógica a ajustar; es decir, el número de términos t que tendría el “modelo de regresión” [1.2]; si este valor es mayor que 1, debemos utilizar también los argumentos **ntrees** (que será igual al valor dado a **select**, y **nleaves** sobre el número de hojas que queremos tenga el árbol ajustado.

Apuntemos de nuevo el hecho de que esta función utiliza métodos bootstrap y que, en consecuencia, cada vez que la ejecutemos obtendremos resultados distintos.

Ejemplo 1.7

Si consideramos de nuevo los datos Virco y la variable dependiente NFV.Fold-IDV.Fold como en los ejemplos anteriores, podemos obtener 1 árbol de regresión lógica ejecutando (1)

```
> library(LogicReg)
> Logico<-logreg(resp=Rasgo22,bin=Virco22,select=1) (1)
```

```
> Logico (3)
score 73.996
-274 * (((P72 or P85) or (not P35)) or (((not P93) or (not P73)) and
((not P25) or P70)))
```

```
> plot(Logico) (2)
```

Ejecutando (2) obtenemos la representación de este árbol dada por la la Figura 1.9. De esta figura, o del resultado de ejecutar (3), vemos que el modelo ajustado es

$$g(E[Y]) = -273'5 \cdot \{[(P72 \vee P85) \vee P35^c] \vee [(P93^c \vee P73^c) \wedge (P25^c \vee P70)]\}$$

Si queremos que el “modelo lineal” tenga dos términos, ejecutaremos (4), obteniendo los resultados con (5) y la Figura 1.10 ejecutando (6) dos veces.

librería `rio` está la función `import` que me va a permitir importar la función `FctMSE`, de la librería `tidyverse`, que calcula los valores del MSE en valores de la red de parámetros y datos en formato data frame. Con (4) incorporamos esta función `FctMSE`.

Con (5) creamos los datos. Mediante (6) calculamos los valores de la función `FctMSE` en los valores de los parámetros de la Red y los datos dados en el enunciado. El argumento `MARGIN=1` se utiliza para indicar que la función `FctMSE` se aplique a las filas en lugar de a las columnas.

Con (7) y (8) buscamos el mínimo de los MSE , y en (9) le pedimos a R que lo muestre. Vemos en (10) que los estimadores de los coeficientes de regresión en los que se obtiene el mínimo MSE son $\hat{\beta}_0 = -75$ $\hat{\beta}_1 = 2$ que difieren de los estimadores de mínimos cuadrados antes calculados, $\hat{\beta}_0 = -77'47$ y $\hat{\beta}_1 = 1'13$.

```
> Beta0Valores=seq(-78,-75,0.1) (1)
```

```
> Beta1Valores=seq(1,2,0.1) (2)
```

```
> Red=expand.grid(Beta0=Beta0Valores,Beta1=Beta1Valores) (3)
```

```
> library(tidyverse)
```

```
> library(rio)
```

```
> FctMSE=import("https://ai.lange-analytics.com/source/FctMSE.rds") (4)
```

```
> Edad<-c(67.5,72.5,77.5,82.5,87.5,92.5) (5)
```

```
> Prevalencia<-c(3,4,8,14,20,32) (5)
```

```
> datos<-data.frame(Edad,Prevalencia) (5)
```

```
> salida=apply(Red,FUN=FctMSE,data=datos,MARGIN=1) (6)
```

```
> parametrosdatos=mutate(Red,MSE=salida) (7)
```

```
> solucion=filter(parametrosdatos,MSE==min(MSE)) (8)
```

```
> print(solucion) (9)
```

```
  Beta0 Beta1    MSE
1   -75     2 1778732 (10)
```

Si aumentamos el tamaño de la Red podríamos obtener más precisión.

3.3. Tamaño muestral muy elevado

Si el tamaño muestral n es muy grande, dado que los estimadores de regresión tienen un error de muestreo en donde n figura en el denominador, si éste es muy grande, tenderá a cero el error de muestreo y todas las covariables consideradas van a resultar significativas.

3.4. Sobre-ajuste

Sabemos que una recta pasa exactamente por dos puntos; también que un polinomio de grado 2 pasa exactamente por 3 puntos. Si queremos, siempre podemos encontrar un polinomio de grado $n - 1$ que pase exactamente por n puntos. De esta forma, el error MSR es 0 porque el polinomio pasa exactamente por todos los puntos, pero esa función (ese polinomio) es tan especializado que si salimos del conjunto de puntos para el que lo hemos calculado (muestra de entrenamiento), seguramente predecirá muy mal a otro conjunto de puntos resultado de la observación de las mismas variables.

Es decir, cuando ajustamos una función a un conjunto de puntos queremos que esté especializada en ese conjunto de puntos pero que nos sirva para otro conjunto de puntos, es decir, que haya un balance entre esos dos objetivos y que no haya lo que se denomina sobre-ajuste (*overfitting*). Un polinomio con grado pequeño será más flexible y es más deseable para evitar el sobre-ajuste. Este objetivo se consigue ajustando una función mediante Inteligencia Artificial. Si la muestra de entrenamiento tiene un tamaño pequeño, el problema de sobre-ajuste se agrava.

El problema del sobre-ajuste se puede resolver con 1) Regularización (Lasso, Ridge y Elastic Net) que se estudia en el siguiente capítulo, o bien con 2) Validación Cruzada (K-fold Cross-Validation) (véase EAAR-sección 18.2). Esta técnica permite evaluar el rendimiento del modelo en diferentes subconjuntos de datos para asegurar que no esté “memorizando” la muestra de entrenamiento; se pueden usar las librerías `caret` o `tidymodels` (más abajo vemos un ejemplo) o con, 3) Selección de Variables y Simplificación. Un modelo con demasiadas variables predictoras es más propenso al overfitting. Stepwise Regression: Selección automática basada en criterios como el AIC o BIC mediante la función `step()`. Reducción de dimensionalidad: Utilizar PCA (Análisis de Componentes Principales) EAAR-capítulo 2, para resumir la información en menos variables.

O, por último, con 4) Técnicas para Modelos Específicos utilizando Árboles de Decisión: Aplicar poda (pruning) para limitar la profundidad del árbol y evitar que aprenda casos demasiado específicos. Random Forest: Ajustar el hiperparámetro `mtry` (número de variables candidatas en cada división) mediante validación cruzada. Redes Neuronales (Keras/TensorFlow en R): Utilizar capas de Dropout (desactivación aleatoria de neuronas) o Early Stopping (detener el entrenamiento cuando el error de validación deja de bajar).

Ejemplo 3.2

Los datos `Casas.txt` (Kaggle, 2015) corresponden a 21.613 viviendas en las que observamos varias variables (King County House Sale dataset). Una de estas variables es el Precio de la Casa, `price`, y otra la Superficie habitable, `sqft-living`.

Con (1) incorporamos los datos a R. Después dividimos la muestra en los conjuntos Muestra

Capítulo 4

Selección de Predictores: Regresión Lasso, Ridge y Elastic Net

4.1. Introducción

En este capítulo utilizamos la Regularización como técnica alternativa para evitar el sobre-ajuste. En la esencia de esta técnica de Regularización está el introducir una *penalización* para controlar el número de parámetros a incorporar en el modelo.

Veremos las penalizaciones más habituales que son Lasso, Ridge y Elastic Net. La tercera es combinación de las dos primeras. Puede verse EAA-capítulo 18.

Como en el capítulo anterior, trataremos de minimizar el *Mean Square Error*

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - y_{t_i})^2$$

en donde

$$y_{t_i} = \widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \dots + \widehat{\beta}_k X_k.$$

Ahora incluiremos en la función objetivo una penalización, tratando de minimizar

$$\frac{1}{n} \sum_{i=1}^n (y_i - y_{t_i})^2 + \lambda \sum_{j=1}^k |\beta_j|$$

en el caso de Regresión Lasso, y minimizaremos

$$\frac{1}{n} \sum_{i=1}^n (y_i - y_{t_i})^2 + \lambda \sum_{j=1}^k \beta_j^2$$

en el caso de Regresión Ridge. Por último, minimizaremos

$$\frac{1}{n} \sum_{i=1}^n (y_i - y_{t_i})^2 + \lambda \left[\frac{(1 - \alpha)}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right]$$

en el caso de Elastic Net, en donde $\|\beta\|_1 = \sum_{j=1}^k |\beta_j|$ es la denominada *Norma* L_1 y $\|\beta\|_2 = \sqrt{\sum_{j=1}^k \beta_j^2}$ la norma L_2 o Euclídea.

4.2. Regresión Ridge

La Regresión Ridge es una técnica de modelado estadístico y aprendizaje automático utilizada para analizar datos que sufren de multicolinealidad (altas correlaciones entre predictores) o cuando existe un alto riesgo de sobre-ajuste (overfitting) en un modelo de regresión lineal simple.

Con la Regresión Ridge reducimos la varianza del modelo, a costa de introducir un pequeño sesgo (*bias*). Ridge contrae (reduce) los coeficientes de regresión hacia cero, pero nunca los hace exactamente cero. Esto significa que mantiene todas las variables como significativas, pero disminuye el impacto de las menos importantes.

La suma residual de cuadrados, o Suma de Cuadrados de los Errores (*SSE*), también denominada Suma de Cuadrados no Explicada SSNEX en EBR-sección 10.3.1,

$$SSE = \sum_{i=1}^n (y_i - y_{t_i})^2$$

puede ser 0 y, en ese caso, el modelo predice perfectamente las variables dependientes. Sin embargo, esto no siempre es deseable, ya que puede indicar un exceso de ajuste en los datos, especialmente si estos son pocos. La multicolinealidad puede ser una de las causas.

Las estimaciones de coeficientes altos a menudo pueden ser síntomas de sobre-ajuste. Si dos o más variables comparten una correlación lineal alta, el estimador de mínimos cuadrados puede conducir a coeficientes de valor alto erróneos. Cuando uno o más coeficientes son demasiado altos, la salida del modelo se vuelve sensible a alteraciones menores en los datos de entrada. En otras palabras, el modelo se ha sobre-ajustado en un conjunto de entrenamiento específico y no puede generalizar con precisión en nuevos conjuntos de prueba. Este modelo se considera inestable.

El parámetro λ se denomina parámetro de regularización y controla la magnitud de la penalización. Si $\lambda = 0$, Ridge es idéntico a una regresión lineal ordinaria. Si $\lambda \rightarrow \infty$, los coeficientes se aproximan a cero.

Para ajustar un modelo Ridge con R debemos utilizar la función

```
glmnet(x, y, alpha = 0)
```

de la librería `glmnet`. Los argumentos mínimos de esta función son: **x** covariables independientes en formato matriz; **y** es la variable dependiente; por último, el argumento **alpha**, que si le hacemos tomar el valor 0, estamos en un caso de regresión Ridge; si toma el valor 1 estamos en un caso de regresión Lasso y, en general, ponemos el valor

$$\frac{(1 - \alpha)}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1$$

en el caso de Elastic Net, en donde $\|\beta\|_1 = \sum_{i=1}^k |\beta_j|$ es la denominada *Norma* L_1 y $\|\beta\|_2 = \sqrt{\sum_{i=1}^k \beta_j^2}$ la norma L_2 o Euclídea.

Ejemplo 4.1

Vamos a utilizar los datos `mtcars` de la librería `datasets`, Henderson y Velleman (1981), consistente en pruebas de carretera de automóviles considerados por la revista estadounidense Motor Trend de 1974, e incluyen el consumo de combustible y 10 aspectos del diseño y el rendimiento de 32 automóviles. En concreto se consideraron las variables: **mpg** Millas por galón, **cyl** Número de cilindros, **disp** Desplazamiento (en pulgadas cúbicas), **hp** Caballos de fuerza, **drat** Relación del eje trasero, **wt** Peso (1000 libras), **qsec** 1/4 tiempo obtenido en 1 milla, **vs** motor (0 = en forma de V, 1 = derecho), **am** Transmisión (0 = automático, 1 = manual), **gear** Número de marchas para delante, **carb** Número de carburadores.

Abrimos la librería de datos con (1) y los incorporamos en (2). En (3) definimos las covariables independientes y la dependiente. En (4) ejecutamos la regresión Ridge. Es necesario estandarizar los datos para que no haya una variable que domine la regresión, pero R estandariza siempre que pongamos el valor 0 para α .

Para ver cómo cambian los estimadores de los coeficientes de regresión en función de α , ejecutamos (5).

Para elegir el mejor modelo ejecutamos (6).

Elegimos el mejor valor de λ mediante Validación Cruzada, con (6). El parámetro λ controla la fuerza de la penalización. Un alto significa más penalización (menos sobreajuste, pero más sesgo). Con el valor de λ así elegido, determinamos los estimadores de los coeficientes con (7). Recordemos al lector que con el método de Validación Cruzada, cada vez que lo ejecute puede obtener un resultado distinto con lo que los coeficientes obtenidos en (7) pueden ser diferentes.

```
> library(datasets) (1)
```

```
> data("mtcars") (2)
```

```
> x<-as.matrix(mtcars[,c("cyl","disp","hp","drat","wt","qsec")]) (3)
```

```
> y<-mtcars$mpg (3)
```