



INFERENCIA Y MODELOS ESTADÍSTICOS

Jacqueline Köhler C. y José Luis Jara V.





CAPÍTULO 15. REGRESIÓN LOGÍSTICA

En los capítulos 13 y 14 estudiamos la regresión lineal, útil para predecir una respuesta numérica a partir de una o más variables, aunque con una serie de condiciones. Como explican Field et al. (2012, pp. 312-345), la **regresión logística** es un ejemplo de los **modelos lineales generalizados**, los que admiten una variable respuesta cuyos residuos sigan una distribución diferente a la normal.

La regresión logística relaciona la distribución de la variable de respuesta con un modelo lineal usando como función de enlace la **función logística estándar**, implementada —por ejemplo— como `logistic()` en el paquete `psych` de R, que presentamos en la ecuación 15.1 y mostramos gráficamente en la figura 15.1. Esta función describe una **transición de cero a uno**, asociados a valores estandarizados z , por lo que resulta especialmente útil para representar la **probabilidad** de que ocurra algún evento: un valor cercano a cero indica que es muy poco probable, mientras un valor cercano a 1 corresponde a una alta probabilidad (lógicamente, un valor de 0,5 indica que es igualmente probable que el evento ocurra o no). Así, la regresión logística resulta adecuada para predecir una respuesta dicotómica, que puede ser asociada a una **distribución binomial**.

$$P(z) = \frac{1}{1 + e^{-z}} \quad (15.1)$$

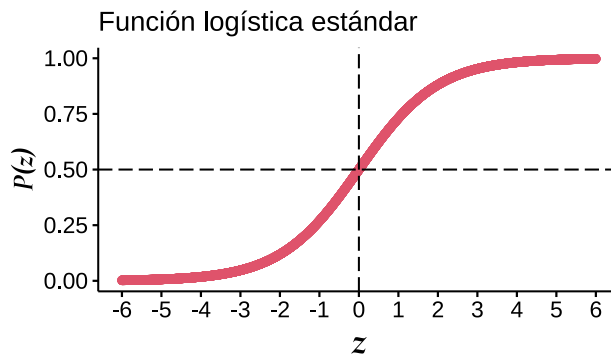


Figura 15.1: función logística estándar.

Para entender mejor esta idea, necesitamos introducir el concepto de **odds** (Cerde et al., 2013), el cual no tiene una traducción directa al castellano, pero que puede entenderse como “oportunidad” o “chance”, aunque a veces se traduce incorrectamente como “probabilidad”. Matemáticamente, los **odds** está dado por la ecuación 15.2, por lo que se define como la razón entre la probabilidad de que ocurra un evento, y la probabilidad de que este no ocurra.

$$odds(p) = \frac{p}{1 - p} \quad (15.2)$$

Tomemos el ejemplo que usan Cerde et al. (2013): supongamos que los registros históricos dicen que en junio llueve 12 días. Así, la probabilidad de que llueva un día de junio es:

$$\Pr(\text{día de lluvia en junio}) = \frac{12}{30} = 0,40$$

Pero la oportunidad de que el día sea lluvioso, equivalente a la expresión en la ecuación 15.2, es:

$$odds(\text{día de lluvia en junio}) = \frac{12}{18} = 0,67$$

Ambas medidas presentan la misma información, pero de manera diferente.

Notemos que, en general, cuando un evento e tiene las mismas posibilidades de ocurrir, $\Pr(Y = e) = \Pr(Y = \neg e) = 0,5$ y $odds(Y = e) = 1$.

De acuerdo a estas definiciones, el logaritmo de los *odds*, llamada **función logit** o *log-odds* en contextos angloparlantes, sigue una distribución normal que corresponde a la función inversa de la logística estándar. Esta función está implementada en varios paquetes de R como `logit()`, incluyendo el mencionado `psych`. Así, podemos relacionar los *odds* y las probabilidades como muestran las ecuaciones 15.3 y 15.4.

$$z = \text{logit}(p) = \log\left(\frac{p}{1-p}\right) \quad (15.3)$$

$$p = \text{logistic}(z) = \frac{1}{1 + e^{-z}} \quad (15.4)$$

A su vez, podemos asociar z a otras variables numéricas $X = (x_1, x_2, \dots, x_n)$ por medio de una combinación lineal con coeficientes $B = (\beta_0, \beta_1, \dots, \beta_n)$, como muestra la ecuación 15.5.

$$z = X \odot B = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n \quad (15.5)$$

Así, la regresión logística nos permite **enlazar** un conjunto de variables predictoras a la probabilidad de ocurrencia de un evento e a través de la combinación lineal $X \odot B$, de acuerdo a la ecuación 15.6.

$$\Pr(Y = e|X; B) = \text{logistic}(X \odot B) = \frac{1}{1 + e^{-(X \odot B)}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}} \quad (15.6)$$

De esta forma podemos seleccionar una división o **umbral** que permita **predecir** la ocurrencia de un evento e o, de forma equivalente, **clasificar** un objeto en dos categorías posibles:

- Para valores menores que el umbral se predice que el evento e “no ocurre”, otorgándose una clasificación cero ($\hat{y} = 0$) o negativa ($\hat{y} = -$).
- Mientras que para valores mayores o iguales que el umbral se predice que el evento “sí ocurre”, a lo que corresponde una clasificación uno ($\hat{y} = 1$) o positiva ($\hat{y} = +$).

Si bien es usual utilizar el valor $\Pr(Y = 1) = 0,5$ como umbral, esto no es obligatorio ni siempre conveniente, como veremos más adelante.

En capítulos precedentes explicamos que el ajuste de un modelo de regresión lineal se realiza mediante la resolución de un problema de optimización que busca minimizar la suma de las desviaciones cuadradas entre las respuestas predichas y las observadas. En el caso de la regresión logística, también se ajusta mediante la resolución de un problema de optimización, donde buscamos minimizar la diferencia entre las respuestas observadas y las respuestas predichas. Como las respuestas corresponden a una **variable dicotómica**, esta optimización se realiza usando la función de verosimilitud, aunque, por conveniencia, se suele optimizar el logaritmo natural de la verosimilitud. Si se tiene una muestra $D = \{(X_j, y_j)\}$ con d observaciones —donde $j = 1, 2, \dots, d$; cada $y_j \in \{0, 1\}$; y $X_j = (x_{j1}, x_{j2}, \dots, x_{jn})$ son los valores observados de las variables predictoras en conjunto a y_j —, la función de verosimilitud a optimizar en búsqueda del vector de parámetros $B = (\beta_0, \beta_1, \dots, \beta_n)$ es:

$$\begin{aligned} \mathcal{L}(B; D) &= P(y_1, y_2, \dots, y_d | X_1, X_2, \dots, X_d; B) \\ &= \prod_{j=1}^d \Pr(y_j = 1 | X_j; B)^{y_j} (1 - \Pr(y_j = 1 | X_j; B))^{1-y_j} \\ &= \prod_{j=1}^d \text{logistic}(X_j \odot B)^{y_j} (1 - \text{logistic}(X_j \odot B))^{1-y_j} \end{aligned} \quad (15.7)$$

Aplicando logaritmo natural se obtiene:

$$\ln \mathcal{L}(B; D) = \sum_{j=1}^d [y_j \ln(\text{logistic}(X_j \odot B)) + (1 - y_j) \ln(1 - \text{logistic}(X_j \odot B))] \quad (15.8)$$



15.1 EVALUACIÓN DE UN CLASIFICADOR

Una forma de evaluar modelos de clasificación, entre ellos los de regresión logística, es de acuerdo a la cantidad de errores cometidos (Zelada, 2017). Para ello, el primer paso consiste en construir una tabla de contingencia (también llamada matriz de confusión) para las respuestas predichas y observadas, como muestra la tabla 15.1, bastante similar a la que ya conocimos para explicar los errores de decisión en la prueba de hipótesis (tabla 4.1). Las cuatro celdas de la matriz de confusión contienen:

- **Verdaderos positivos** (VP): cantidad de instancias correctamente clasificadas como pertenecientes a la clase positiva.
- **Falsos positivos** (FP): cantidad de instancias erróneamente clasificadas como pertenecientes a la clase positiva.
- **Falsos negativos** (FN): cantidad de instancias erróneamente clasificadas como pertenecientes a la clase negativa.
- **Verdaderos negativos** (VN): cantidad de instancias correctamente clasificadas como pertenecientes a la clase negativa.

		Real		Total
		1 (+)	0 (-)	
Clasificación	1 (+)	VP	FP	$VP + FP$
	0 (-)	FN	VN	$FN + VN$
	Total	$VP + FN$	$FP + VN$	n

Tabla 15.1: tabla de contingencia para evaluar un clasificador.

La **exactitud** (*accuracy*) del modelo corresponde a la proporción de observaciones correctamente clasificadas, dada por la ecuación 15.9.

$$\text{exactitud} = \frac{VP + VN}{n} \quad (15.9)$$

A su vez, el **error** del modelo corresponde a la proporción de observaciones clasificadas de manera equivocada (ecuación 15.10).

$$\text{error} = \frac{FP + FN}{n} = 1 - \text{exactitud} \quad (15.10)$$

La **sensibilidad** (*sensitivity* o *recall*, ecuación 15.11) indica cuán apto es el modelo para detectar aquellas observaciones pertenecientes a la clase positiva.

$$\text{sensibilidad} = \frac{VP}{VP + FN} \quad (15.11)$$

De manera análoga, la **especificidad** (*specificity*, ecuación 15.12) permite determinar cuán exacta es la asignación de elementos a la clase positiva. También puede entenderse como la aptitud del modelo para correctamente asignar observaciones a la clase negativa.

$$\text{especificidad} = \frac{VN}{FP + VN} \quad (15.12)$$

La **precisión** (*precision*) o valor predictivo positivo (*VPP*, ecuación 15.13) indica la proporción de instancias clasificadas como positivas que realmente lo son.

$$VPP = \frac{VP}{VP + FP} \quad (15.13)$$

Asimismo, el **valor predictivo negativo** (*VPN*, ecuación 15.14) señala la proporción de instancias correctamente clasificadas como pertenecientes a la clase negativa.



$$VPN = \frac{VN}{FN + VN} \quad (15.14)$$

Otra herramienta útil es la **curva de calibración**, también llamada curva ROC por las siglas inglesas para *receiver-operating characteristic*, que muestra la relación entre la sensibilidad y la especificidad del modelo (Glen, 2017). Este gráfico también permite evaluar la precisión del modelo, puesto que mientras más se aleje la curva de la diagonal, mayor es la precisión. Para ilustrar mejor la utilidad de este gráfico, la figura 15.2 muestra las curvas ROC para dos modelos diferentes, además de la diagonal. Esta figura indica que el clasificador representado por la curva morada es mejor que el representado por la curva azul, pues se aleja más de la diagonal. Este “alejamiento” de la diagonal, a veces es representado numéricamente por el área total bajo la curva ROC, llamado AUC por sus inglesas para *area under the curve*, cuyo valor varía entre 0 y 1. Un AUC más alto indica un mejor desempeño del modelo en la clasificación. Un modelo perfecto que clasifica correctamente todas las instancias exhibe un $AUC = 1$, mientras que un modelo que no discrimina, es decir, su desempeño no es mejor que el de una clasificación aleatoria, se asocia a un $AUC = 0,5$.

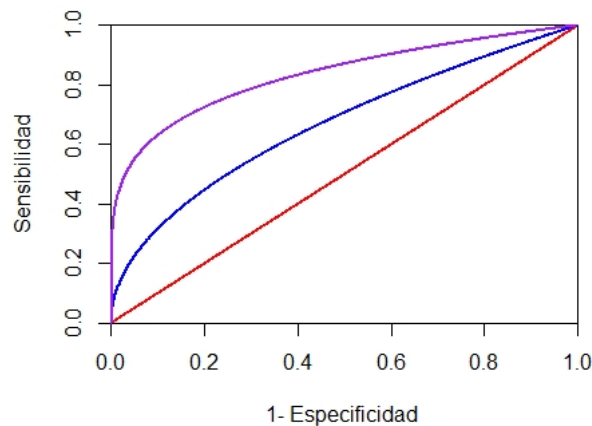



Figura 15.2: dos curvas ROC. Fuente: Ayala (2020).

15.2 CONDICIONES PARA USAR REGRESIÓN LOGÍSTICA

Desde luego, no basta con evaluar el desempeño del clasificador, sino que también necesitamos verificar el cumplimiento de ciertas condiciones para que un modelo de regresión logística sea válido:

1. Debe existir una relación lineal entre los predictores y la respuesta transformada.
2. Los residuos deben ser independientes entre sí.

Además de las condiciones anteriores, existen otras situaciones en que puede ocurrir que el método de optimización no converja:

1. Multicolinealidad entre los predictores, que en este caso se aborda del mismo modo que para RLM (por ejemplo, mediante el factor de inflación de la varianza o la tolerancia).
2.  Información incompleta, que se produce cuando no contamos con observaciones suficientes para todas las posibles combinaciones de predictores.
3. Separación perfecta, que ocurre cuando no hay superposición entre las clases, es decir, ¡cuando los predictores separan ambas clases completamente!



15.3 BONDAD DE AJUSTE DEL MODELO

Al igual que en el caso de la regresión lineal, existen diversos mecanismos para evaluar el ajuste de un modelo de regresión logística. El estadístico de **log-verosimilitud** ($\ln \mathcal{L}(B; D)$), dado por la ecuación 15.8, nos permite cuantificar la diferencia entre las probabilidades predichas y las observadas. Este estadístico se asemeja a la suma de los residuos cuadrados de la regresión lineal en el sentido de que cuantifica la cantidad de información que carece de explicación tras el ajuste del modelo. Así, mientras menor sea su valor, mejor es el ajuste del modelo.

Sin embargo, el estadístico **desviación** (*deviance* en inglés), a menudo denotada por $-2LL$ y en pocas ocasiones traducida como *devianza*, suele usarse en lugar de la log-verosimilitud, y que se obtiene de forma directa a partir de esta última, como indica la ecuación 15.15.

$$-2LL = -2 \ln \mathcal{L}(B; D) \quad (15.15)$$



Al igual que con los modelos de regresión lineal, podemos comparar modelos de regresión logística mediante la función `anova()`, aunque ahora la prueba F resulta inapropiada. Si queremos comparar dos modelos M_1 y M_2 , en que M_2 contiene todos los predictores de M_1 más otros k predictores, entonces la diferencia de sus desviaciones, $(-2LL_1) - (-2LL_2)$, **sigue asintóticamente** (a medida que el tamaño de la muestra crece al infinito) una distribución $\chi^2(k)$. Esto permite calcular el nivel de significación de esta diferencia, que es usada en la prueba conocida como *Likelihood Ratio Test* (LRT).

Por otro lado, los criterios de evaluación de modelos basados en el principio de parsimonia, que estudiamos en el capítulo 14, también están definidos para la regresión logística. El más sencillo es el criterio de información de Akaike (AIC), dado por la ecuación 15.16, donde k corresponde a la cantidad de predictores en el modelo.

$$AIC = -2LL + 2k \quad (15.16)$$

Similar al AIC, el criterio bayesiano de Schwarz (BIC) ajusta la penalización a la complejidad del modelo según el tamaño de la muestra, como se ve en la ecuación 15.17.

$$BIC = -2LL + 2k \cdot \ln n \quad (15.17)$$

15.4 GENERALIDAD DEL MODELO

En el capítulo anterior vimos que teníamos que examinar los residuos para identificar casos donde el modelo no se ajusta bien o que ejercen una influencia indebida en el modelo. Para evaluar lo primero, examinamos si existían valores atípicos en los residuos “estandarizados” y “studentizados”. Para evaluar lo segundo, usamos estadísticas de influencia como la distancia de Cook, el DFBeta y las estadísticas de apalancamiento. Este procedimiento **aplica de la misma manera** en la regresión logística, por lo que no entraremos más detalles.

15.5 REGRESIÓN LOGÍSTICA EN R

En R, la llamada `glm(formula, family = binomial(link = "logit"), data)` permite ajustar un modelo de regresión logística, donde:

- `formula` tiene la forma `<variable de respuesta>~<variables predictoras>`.
- `data`: matriz de datos.

El argumento `family = binomial(link = "logit")` indica que asumiremos una distribución binomial para la variable de respuesta y que usaremos la función logística como enlace. Esto es necesario puesto que existen otros modelos generalizados de regresión lineal.

El script 15.1 muestra un ejemplo de la construcción de un modelo de regresión logística que prediga el tipo de transmisión de un automóvil a partir de su peso, usando para ello el ya conocido conjunto de datos `mtcars` disponible en R (recordemos que podemos consultar la descripción de las variables en la tabla 13.1). Las líneas 7–19 del script (incluyendo comentarios) ilustran la preparación de los datos que se necesita hacer para construir y evaluar apropiadamente el modelo buscado. Primero, se asegura que la variable de salida sea un factor dicotómico, puesto que la variable `am` viene originalmente con valores numéricos (0 = automática, 1 = manual). Luego, se selecciona aleatoriamente un conjunto de entrenamiento, para construir el modelo, el 80 % de las instancias disponibles. Las líneas 22–23 del script muestran el uso de la función `glm()` para ajustar y mostrar en pantalla el modelo deseado. El resultado se muestra en la figura 15.3, donde podemos apreciar que se obtiene una reducción en la desviación, que para este modelo con un predictor (23 grados de libertad) alcanza 12,23 y un criterio de información de Akaike $AIC = 16,23$.

```
Call:
glm(formula = am ~ wt, family = binomial(link = "logit"), data = entrenamiento)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.17498  -0.40172  -0.00176   0.12321   2.26151

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    18.525      8.504   2.178  0.0294 *
wt             -5.883      2.645  -2.224  0.0261 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

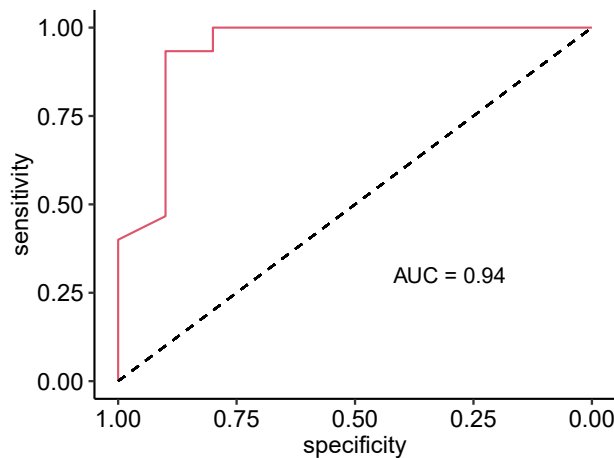
    Null deviance: 34.617  on 24  degrees of freedom
Residual deviance: 12.230  on 23  degrees of freedom
AIC: 16.23

Number of Fisher Scoring iterations: 7
```

Figura 15.3: ajuste de un modelo de regresión logística.

Las líneas 25–41 del script 15.1 evalúan el modelo ajustado usando las herramientas descritas en la sección anterior y el conjunto de entrenamiento. La función `roc(response, predictor)` del paquete `pROC`, donde los argumentos corresponden, respectivamente, a las respuestas observadas para cada instancia y las probabilidades predichas por el modelo de pertenecer la clase positiva (`automática` en este ejemplo), nos permite obtener la curva ROC de la figura 15.4(a). La curva se aleja bastante de la diagonal, por lo que al parecer se trata de un buen clasificador. A su vez, la función `confusionMatrix(data, reference, positive)` del paquete `caret`, donde `data` corresponde a las respuestas predichas, `reference` a las respuestas observadas y `positive` al nombre de la clase positiva, genera la matriz de confusión y obtiene las medidas de evaluación descritas anteriormente, como muestra la figura 15.4(b). Podemos ver que el modelo tiene una exactitud de 92,0 %. La sensibilidad de 100 % y la especificidad de 80,00 % muestran que el modelo se desempeña un poco mejor identificando elementos de la clase positiva, correspondiente en este caso a los vehículos de transmisión automática.

Pero, como ya hemos estudiado en capítulos anteriores, debemos evaluar el modelo con un conjunto de datos



Evaluación del modelo (cjto. de entrenamiento):

Confusion Matrix and Statistics

Prediction	Reference manual	Reference automático
manual	8	0
automático	2	15

Accuracy : 0.92
 95% CI : (0.7397, 0.9902)
 No Information Rate : 0.6
 P-Value [Acc > NIR] : 0.0004293

 Kappa : 0.8276

 McNemar's Test P-Value : 0.4795001

 Sensitivity : 1.0000
 Specificity : 0.8000
 Pos Pred Value : 0.8824
 Neg Pred Value : 1.0000
 Prevalence : 0.6000
 Detection Rate : 0.6000
 Detection Prevalence : 0.6800
 Balanced Accuracy : 0.9000

'Positive' Class : automático

Figura 15.4: evaluación del clasificador en los datos usados para construir el modelo.

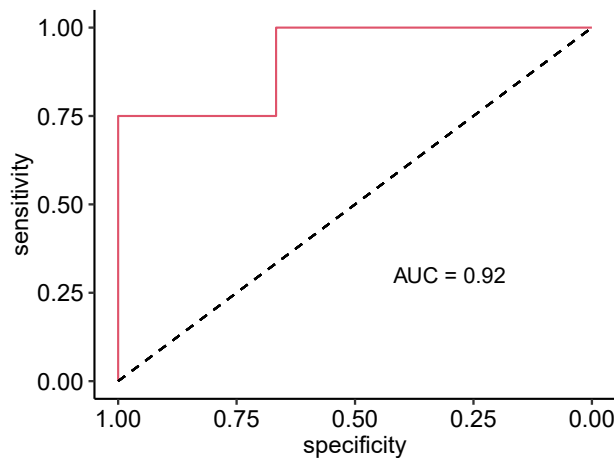
diferente al que usamos para su construcción. Así, las líneas 51–71 obtienen la curva ROC y la matriz de confusión (figuras 15.5 (a) y (b), respectivamente) para el conjunto de prueba, donde observamos un resultado con menor exactitud que con el conjunto de entrenamiento. Esto podría ser una indicación de sobreajuste del modelo al conjunto de entrenamiento, pero también podría ser una señal de que el conjunto de prueba puede ser muy pequeño para obtener una evaluación confiable. Esta última idea gana fuerza si usamos el AUC como criterio de calidad, puesto que los valores son parecidos: 94% con los datos de entrenamiento y 92% con los datos de prueba. Podríamos determinar si estos valores son o no significativamente diferentes usando la función `roc.test()` del mismo paquete `pROC` usando el método de DeLong et al. (1988).

Script 15.1: ejemplo de la construcción de un modelo de regresión logística en R.

```

1 library(caret)
2 library(ggpubr)
3 library(pROC)
4
5 # Cargar los datos.
6 datos <- mtcars
7
8 # Convertir la variable 'am' en factor y cambiar el orden de los
9 # niveles para que el modelo prediga 'automático' (y no 'manual', como
10 # haría por defecto, que es usar el segundo nivel).
11 datos[["am"]] <- factor(datos[["am"]], labels = c("automático", "manual"))
12 datos[["am"]] <- factor(datos[["am"]], levels = c("manual", "automático"))
13
14 # Separar conjuntos de entrenamiento y prueba.
15 set.seed(117)
16 n <- nrow(datos)
17 n_entrenamiento <- floor(0.8 * n)
18 muestra <- sample.int(n = n, size = n_entrenamiento, replace = FALSE)
19 entrenamiento <- datos[muestra, ]
20 prueba <- datos[-muestra, ]
21
22 # Ajustar modelo.

```

Evaluación del modelo (cjto. de prueba):

Confusion Matrix and Statistics

Prediction	Reference manual	Reference automático
manual	3	1
automático	0	3

Accuracy : 0.8571
 95% CI : (0.4213, 0.9964)
 No Information Rate : 0.5714
 P-Value [Acc > NIR] : 0.1243

 Kappa : 0.72

 McNemar's Test P-Value : 1.0000

 Sensitivity : 0.7500
 Specificity : 1.0000
 Pos Pred Value : 1.0000
 Neg Pred Value : 0.7500
 Prevalence : 0.5714
 Detection Rate : 0.4286
 Detection Prevalence : 0.4286
 Balanced Accuracy : 0.8750

'Positive' Class : automático

Figura 15.5: evaluación del clasificador con datos de prueba, no utilizados al construir el modelo.

```

23 modelo <- glm(am ~ wt, family = binomial(link = "logit"), data = entrenamiento)
24 print(summary(modelo))
25
26 # Evaluar el modelo con el conjunto de entrenamiento.
27 probs_e <- predict(modelo, entrenamiento, type = "response")
28
29 # Graficar curva ROC, indicando AUC obtenido.
30 ROC_e <- roc(entrenamiento[["am"]], probs_e)
31 texto_e <- sprintf("AUC = %.2f", ROC_e[["auc"]])
32 g_roc_e <- ggroc(ROC_e, color = 2)
33 g_roc_e <- g_roc_e + geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1),
34                                linetype = "dashed")
35 g_roc_e <- g_roc_e + annotate("text", x = 0.3, y = 0.3, label = texto_e)
36 g_roc_e <- g_roc_e + theme_pubr()
37 print(g_roc_e)
38
39 # Obtener las predicciones
40 umbral <- 0.5
41 preds_e <- sapply(probs_e, function(p) ifelse(p >= umbral, "automático", "manual"))
42 preds_e <- factor(preds_e, levels = levels(datos[["am"]]))
43
44 # Mostrar estadísticas de clasificación
45 cat("\n\nEvaluación del modelo (cjto. de entrenamiento):\n")
46 cat("=====\n")
47 cmat_e <- confusionMatrix(preds_e, entrenamiento[["am"]], positive = "automático")
48 print(cmat_e)
49
50 # Evaluar el modelo con el conjunto de prueba.
51 probs_p <- predict(modelo, prueba, type = "response")
52
53 # Graficar curva ROC, indicando AUC obtenido.
54 ROC_p <- roc(prueba[["am"]], probs_p)
55 texto_p <- sprintf("AUC = %.2f", ROC_p[["auc"]])

```

```

56 g_roc_p <- ggroc(ROC_p, color = 2)
57 g_roc_p <- g_roc_p + geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1),
58                                   linetype = "dashed")
59 g_roc_p <- g_roc_p + annotate("text", x = 0.3, y = 0.3, label = texto_p)
60 g_roc_p <- g_roc_p + theme_pubr()
61 print(g_roc_p)
62
63 # Obtener las predicciones (con el mismo umbral)
64 preds_p <- sapply(probs_p, function(p) ifelse(p >= umbral, "automático", "manual"))
65 preds_p <- factor(preds_p, levels = levels(datos[["am"]]))
66
67 # Mostrar estadísticas de clasificación
68 cat("\n\nEvaluación del modelo (cjto. de prueba):\n")
69 cat("=====\n")
70 cmat_p <- confusionMatrix(preds_p, prueba[["am"]], positive = "automático")
71 print(cmat_p)

```

15.6 EVALUACIÓN MÁS ROBUSTA DE UN CLASIFICADOR

En capítulos anteriores conocimos la validación cruzada como herramienta para conseguir una estimación más confiable y robusta del rendimiento general de un modelo, la cual podemos usar de manera análoga para regresión logística. El script 15.2 mejora el ejercicio realizado en el script 15.1, incorporando el uso de validación cruzada de 5 pliegues. Notemos que la llamada a la función `train()` también solicita que “se guarden” los valores predichos, lo que nos permite estimar el rendimiento promedio del modelo como si se repitiera el script 15.2, seleccionando aleatoriamente un conjunto de entrenamiento y otro de prueba, cinco veces.

Script 15.2: ajuste de un modelo de regresión logística usando validación cruzada.

```

1 library(caret)
2 library(data.table)
3
4 # Cargar los datos.
5 datos <- mtcars
6
7 # Convertir la variable 'am' en factor y cambiar el orden de los
8 # niveles para que el modelo prediga 'automático' (y no 'manual', como
9 # haría por defecto, que es usar el segundo nivel).
10 datos[["am"]] <- factor(datos[["am"]], labels = c("automático", "manual"))
11 datos[["am"]] <- factor(datos[["am"]], levels = c("manual", "automático"))
12
13 # Separar conjuntos de entrenamiento y prueba.
14 set.seed(117)
15 n <- nrow(datos)
16 n_entrenamiento <- floor(0.8 * n)
17 muestra <- sample.int(n = n, size = n_entrenamiento, replace = FALSE)
18 entrenamiento <- datos[muestra, ]
19 prueba <- datos[-muestra, ]
20
21 # Ajustar modelo usando validación cruzada de 5 pliegues.
22 modelo <- train(am ~ wt, data = entrenamiento, method = "glm",
23                family = binomial(link = "logit"),
24                trControl = trainControl(method = "cv", number = 5,
25                                          savePredictions = TRUE))
26 print(summary(modelo))
27
28 # Obtener las predicciones en los datos de prueba
29 umbral <- 0.5
30 probs <- predict(modelo, prueba, type = "prob")

```

```

31 preds <- sapply(probs[["automático"]],
32               function(p) ifelse(p >= umbral, "automático", "manual"))
33 preds <- factor(preds, levels = levels(datos[["am"]]))
34
35 # Mostrar la evaluación del modelo
36 cat("Evaluación del modelo basada en validación cruzada:\n")
37 cat("-----\n")
38 matriz <- confusionMatrix(modelo$pred$pred, modelo$pred$obs)
39 print(matriz)
40
41 cat("Evaluación del modelo en datos de prueba:\n")
42 cat("-----\n")
43 matriz <- confusionMatrix(preds, prueba[["am"]], positive = "automático")
44 print(matriz)
45
46 cat("Detalle por pliegue:\n")
47 cat("-----\n")
48 resumen <- data.table(modelo[["resample"]][, c(1, 3)])
49 resumen <- rbind(resumen, list(modelo[["results"]][[2]], "Mean"))
50 resumen <- rbind(resumen, list(modelo[["results"]][[4]], "SD"))
51 print(resumen[1:5, ], row.names = FALSE)
52 cat("-----\n")
53 print(resumen[6:7, ], row.names = FALSE, col.names = "none")

```

El resultado del script 15.2 puede verse en la figura 15.6. Debemos fijarnos en que el modelo obtenido es idéntico al anterior, ya que la función `train()` reentrena el modelo del pliegue que obtuvo mejor rendimiento con **todos los datos disponibles**. En el caso de la regresión logística (como con la regresión lineal), los pliegues solo se diferencian en los datos que utilizan, manteniendo los predictores, por lo que siempre se llega al mismo modelo. Esto no sería así si la validación cruzada se usara, por ejemplo, para seleccionar las mejores predictoras a incluir en el modelo. En todo caso, el script también ejemplifica cómo ver resultados por pliegue. Puede verse que la función `train()` usó la exactitud para evaluar el modelo obtenido en cada pliegue, y que guarda la media de esta métrica. Este es el comportamiento por defecto, pero se le puede solicitar que considere otras métricas, como la sensibilidad y la especificidad. Incluso se puede definir que se use el AUC de los clasificadores como criterio de selección.

15.7 SELECCIÓN DE PREDICTORES

Cuando tenemos múltiples predictores potenciales, debemos decidir cuáles de ellos incorporar en el modelo. Una vez más, y tal como detallamos en el capítulo 14, el ideal es usar la regresión jerárquica para escoger los predictores de acuerdo a evidencia disponible en la literatura. Sin embargo, al explorar los datos, podemos emplear los demás métodos ya descritos: selección hacia adelante, eliminación hacia atrás, regresión escalonada o todos los subconjuntos. Se usan para ello las mismas funciones de R descritas en el capítulo 14.

15.8 REGRESIÓN LOGÍSTICA EN R CON SELECCIÓN DE PREDICTORES

En páginas previas ajustamos un modelo de regresión logística para determinar el tipo de transmisión de un automóvil a partir de su peso. Sin embargo, el predictor fue seleccionado de manera arbitraria, simplemente para ilustrar el proceso, por lo que podríamos encontrar un mejor modelo usando algún método de selección de predictores. Las líneas 29–41 del script 15.3 llevan a cabo esta tarea usando regresión escalonada, obteniéndose como resultado el modelo presentado en la figura 15.7.

Sin embargo, al ajustar este modelo, R emite algunas advertencias (*warnings*). Para confirmar si esto fue en el proceso de búsqueda o en el modelo encontrado, regeneramos el modelo de forma manual en las líneas 43–46. Lamentablemente, se generan los mensajes que muestra la figura 15.8 sobre problemas con la convergencia del algoritmo que busca los parámetros del modelo. Como se mencionó en la sección 15.2, estas ocurren cuando

```

Call:
NULL

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -19.475      9.120  -2.135  0.0327 *
wt              6.233      2.819   2.211  0.0270 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 33.651  on 24  degrees of freedom
Residual deviance: 12.786  on 23  degrees of freedom
AIC: 16.786

Number of Fisher Scoring iterations: 7

Evaluación del modelo basada en validación cruzada:
-----
Confusion Matrix and Statistics

              Reference
Prediction   manual automático
manual         8           0
automático     2          15

      Accuracy : 0.92
      95% CI   : (0.7397, 0.9902)
No Information Rate : 0.6
P-Value [Acc > NIR] : 0.0004293

      Kappa : 0.8276

McNemar's Test P-Value : 0.4795001

      Sensitivity : 0.8000
      Specificity : 1.0000
      Pos Pred Value : 1.0000
      Neg Pred Value : 0.8824
      Prevalence : 0.4000
      Detection Rate : 0.3200
      Detection Prevalence : 0.3200
      Balanced Accuracy : 0.9000

      'Positive' Class : manual

Evaluación del modelo en datos de prueba:
-----
Confusion Matrix and Statistics

              Reference
Prediction   manual automático
manual         3           1
automático     0           3

      Accuracy : 0.8571
      95% CI   : (0.4213, 0.9964)
No Information Rate : 0.5714
P-Value [Acc > NIR] : 0.1243

      Kappa : 0.72

McNemar's Test P-Value : 1.0000

      Sensitivity : 0.7500
      Specificity : 1.0000
      Pos Pred Value : 1.0000
      Neg Pred Value : 0.7500
      Prevalence : 0.5714
      Detection Rate : 0.4286
      Detection Prevalence : 0.4286
      Balanced Accuracy : 0.8750

      'Positive' Class : automático

Detalle por pliegue:
-----
Accuracy Resample
      1.0 Fold1
      0.8 Fold2
      0.8 Fold3
      1.0 Fold4
      1.0 Fold5
-----
      0.9200000 Mean
      0.1095445 SD

```

Figura 15.6: resultado del script 15.2 que usa validación cruzada.

no hay suficientes datos, los predictores separan completamente las clases, o bien cuando existen problemas de colinealidad.

Al comparar el modelo con el modelo nulo (líneas 48–51), vemos que se reporta una desviación con valor 0. ¡El modelo encontrado no comete errores! Parece que estamos ante el problema que genera una “separación perfecta”. ¿Podría pasar con otras variables? Miremos qué nos dice un indicador de multicolinealidad del modelo completo, estimado en las líneas 53–62 del script y cuyo resultado se muestra en la figura 15.9.

Por regla general, no debería confiarse en modelos con predictores con $VIF \geq 10$. Podemos ver que en este caso se está muy lejos de cumplir esta regla. Parece haber serios problemas con este conjunto de datos, por lo que muy probablemente sea imposible encontrar un modelo con búsqueda escalonada de forma automática.

Para empezar una búsqueda manual, veamos qué pasa con solo un paso adelante del modelo nulo, como se hace en las líneas 64–68 del script. El resultado se aprecia en la figura 15.10.

Podemos ver que algunas variables producirían aumentos del AIC (`hp`, `qseq`, `carb`). La variable con mayor potencial parece ser el peso del vehículo (`wt`). El script crea este primer modelo en las líneas 70–71 y se actualiza el modelo completo eliminando las variables inútiles (líneas 74–75). Podemos avanzar y analizar si

```

Modelo con regresión escalonada
-----
Call:
glm(formula = am ~ wt + hp, family = binomial(link = "logit"),
    data = entrenamiento)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.182e+02  3.646e+05  -0.001    0.999
wt           1.561e+02  1.337e+05   0.001    0.999
hp          -4.801e-01  4.762e+02  -0.001    0.999

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 3.3651e+01  on 24  degrees of freedom
Residual deviance: 2.1309e-09  on 22  degrees of freedom
AIC: 6

Number of Fisher Scoring iterations: 25

```

Figura 15.7: modelo de regresión logística obtenido mediante regresión escalonada.

```

Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred

```

Figura 15.8: modelo de regresión logística obtenido mediante regresión escalonada.

podemos agregar otro predictor, como se hace en las líneas 77–81. El resultado aparece en la figura 15.11.

Una vez más podemos descartar variables porque producirían aumentos del AIC o separación perfecta (`disp`, `drat`, `gear`), y escoger el rendimiento vehículo (`mpg`) como predictor más prometedor. El script crea este modelo en las líneas 83–84 y actualiza el modelo completo en las líneas 86–87. Al revisar si podemos avanzar a un tercer predictor (líneas 77–81), observamos que ambas variables disponibles producen un aumento del AIC, por lo que detenemos la búsqueda.

Hemos conseguido dos modelos, uno que utiliza el peso del vehículo y otro que, además, considera su rendimiento (`wt`). Las líneas 98–102 del script 15.3 comparan estos modelos usando la prueba LRT mencionada previamente, cuyo resultado puede verse en la figura 15.12. Vemos que se consiguen reducciones significativa de desviación, primero, con el primer predictor ($\chi^2(1) = 20,86, p < 0,001$) y, luego, al agregar el segundo ($\chi^2(1) = 5,56, p = 0,018$).

```

Multicolinealidad modelo completo
-----
VIF:
      mpg      cyl      disp      hp      drat      wt      qsec
26.30712 560.89915 141.47841 58.00055 28.21426 91.11566 11.30808
      vs      gear      carb
46.08075 373.81477 3025.88396

VIF promedio: [1] 436.3103

```

Figura 15.9: factores de inflación de la varianza para el modelo completo.

```

Single term additions

Model:
am ~ 1
      Df Deviance    AIC
<none>      33.651 35.651
mpg      1   24.893 28.893
cyl      2   26.541 32.541
disp     1   22.953 26.953
hp       1   32.957 36.957
drat     1   16.957 20.957
wt       1   12.786 16.786
qsec     1   32.282 36.282
vs       1   31.910 35.910
gear     2   11.457 17.457
carb     5   26.563 38.563

```

Figura 15.10: información para escoger el primer predictor.

```

Segundo paso
-----
Single term additions

Model:
am ~ wt
      Df Deviance    AIC
<none>      12.7860 16.786
mpg      1    7.2265 13.226
cyl      2    8.1946 16.195
disp     1   10.9637 16.964
drat     1   11.0068 17.007
vs       1    7.7423 13.742
gear     2    0.0000  8.000

```

Figura 15.11: información para escoger el segundo predictor.

De esta forma, seguiremos con el mejor modelo encontrado, el que usa dos predictores. Verifiquemos que no exista multicolinealidad fuerte, con el código de las líneas 104–113. Este nos reporta iguales factores de inflación de varianza ($VIF = 6,85$) para ambos predictores. Si bien son un poco altos, pues hay autores que sugieren que valores superiores a 3 ‘pueden ser problemáticos, no sobrepasan el umbral crítico de 10.

Para verificar el supuesto de linealidad entre los predictores y la respuesta transformada, las líneas 115–125 del script generan los gráficos de dispersión respectivos, que se muestran en la figura 15.13, donde podemos apreciar una relación lineal clara con el peso del vehículo (`wt`) y con algo de dispersión con el rendimiento (`mpg`).

Las líneas 127–131 del script 15.3 verifican la independencia de los residuos por medio de la prueba de Durbin-Watson, la que resulta no significativa ($DW = 2,18, p = 0,584$), por lo que no hay motivos para sospechar que exista una dependencia serial en el modelo.

Para verificar (visualmente) la normalidad de los residuos, las líneas 134–140 del script obtienen un gráfico cuartil-cuartil de ellos, marcando aquellos que estén en los extremos. Podemos ver este gráfico en la figura 15.14, donde se aprecia que los residuos siguen bien una distribución normal, con solo dos casos que caen fuera del 95 % de ella.

Falta determinar si el ajuste del modelo seleccionado se ve afectado por la presencia de valores atípicos.

```

Likelihood Ratio Test para los modelos
-----
Analysis of Deviance Table

Model 1: am ~ 1
Model 2: am ~ wt
Model 3: am ~ wt + mpg
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1         24      33.651
2         23      12.786  1   20.8646 4.929e-06 ***
3         22       7.227  1    5.5595 0.01838 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Figura 15.12: comparación de los modelos con un único predictor.

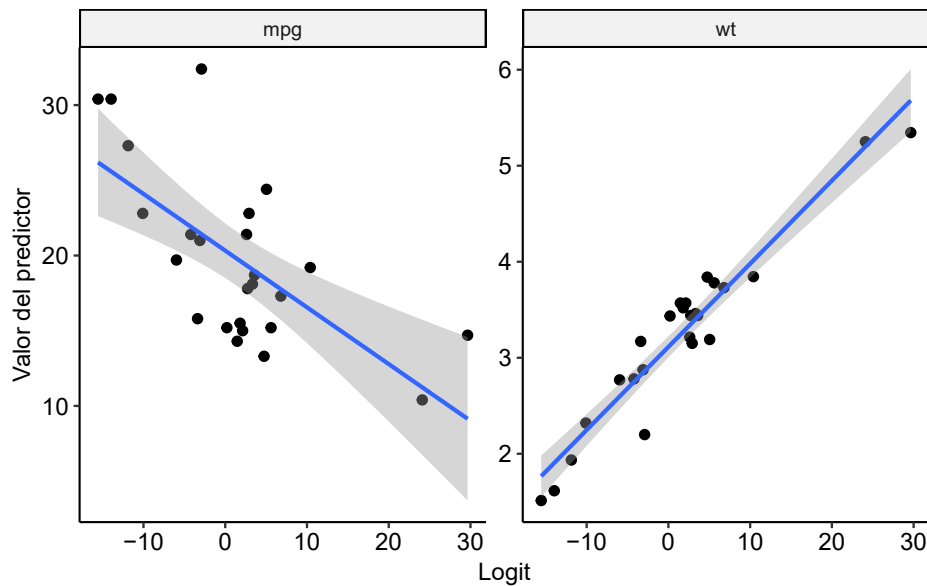


Figura 15.13: gráficos de dispersión entre los predictores y la respuesta transformada.

En las líneas 134–139 del script 15.3 se obtienen y muestran las estadísticas de influencia del modelo, cuyo resultado se muestra en la figura 15.15. En ella, podemos observar que se reconocen cuatro casos demasiado influyentes, según dos siguientes criterios:

1. $|\text{dfb.x}| > 1$, donde dfb.x representa la diferencia en betas (DFBeta) de cada parámetro \mathbf{x} del modelo, es decir, el intercepto y los predictores.
2. $|\text{ddfit}| > 3\sqrt{\frac{k+1}{n-k-1}}$, siendo ddfit la diferencia en ajuste (DFFit), n es el tamaño de la muestra y k es el número de predictores considerados por el modelo.
3. $|1 - \text{cov.r}| > 3\frac{k+1}{n-k-1}$, donde cov.r es la razón de covarianza.
4. $\Pr_{F(k+1, n-k-1)}(|\text{cook.d}|) > 0,5$, donde cook.d es la distancia de Cook.
5. $|\text{hat}| > 3\frac{k+1}{n}$, donde hat es el apalancamiento.

En consecuencia, Maserati Bora no cumple los criterios 2 y 3; AMC Javelin falla los criterios 1, 2 y 5; Fiat 128 no pasa los criterios 2, 3 y 5; y finalmente el Ford Pantera L no cumple el criterio 3, aunque está en el borde ($|1 - 1,416| = 0,416 > 0,409$).

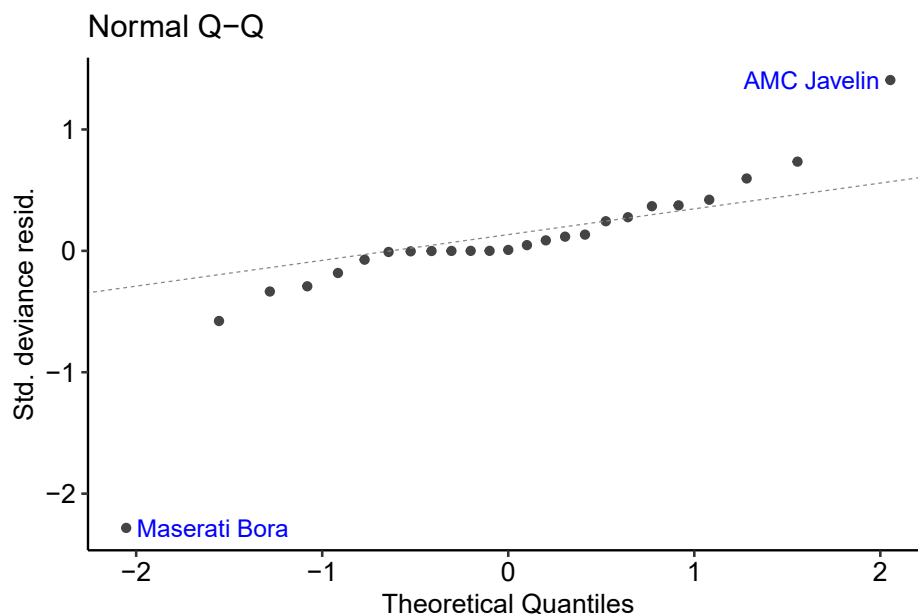


Figura 15.14: gráfico cuartil-cuartil de los residuos estandarizados.

Correspondería entonces revisar si los valores registrados para estos casos no tienen errores que deban corregirse. Si esto se confirma, debemos entablar una discusión con los dueños de los datos, y expertos en el negocio, sobre la posible eliminación de estos casos. Por ahora, dejamos esta búsqueda hasta aquí. Por supuesto, falta evaluar el poder predictivo del modelo, pero esto es idéntico a lo realizado en el script 15.1 o, más avanzadamente, el script 15.2.

Script 15.3: ajuste y evaluación del mejor modelo para predecir el tipo de transmisión de un automóvil.

```

1 library(car)
2 library(ggfortify)
3 library(ggpubr)
4 library(psych)
5 library(tidyverse)
6
7 # Cargar los datos.
8 datos <- mtcars
9
10 # Convertir la variable 'am' en factor y cambiar el orden de los
11 # niveles para que el modelo prediga 'automático' (y no 'manual', como
12 # haría por defecto, que es usar el segundo nivel).
13 datos[["am"]] <- factor(datos[["am"]], labels = c("automático", "manual"))
14 datos[["am"]] <- factor(datos[["am"]], levels = c("manual", "automático"))
15
16 # Sincerar las otras variables no numéricas
17 datos[["cyl"]] <- factor(datos[["cyl"]])
18 datos[["vs"]] <- factor(datos[["vs"]])
19 datos[["gear"]] <- factor(datos[["gear"]])
20 datos[["carb"]] <- factor(datos[["carb"]])
21
22 # Separar conjuntos de entrenamiento y prueba
23 set.seed(117)
24 n <- nrow(datos)
25 n_entrenamiento <- floor(0.8 * n)
26 muestra <- sample.int(n = n, size = n_entrenamiento, replace = FALSE)
27 entrenamiento <- datos[muestra, ]
28 prueba <- datos[-muestra, ]

```


Casos sospechosos de apalancamiento

Influence measures of

```
glm(formula = am ~ wt + mpg, family = binomial(link = "logit"), data = entrenamiento) :
```

	dfb.1_	dfb.wt	dfb.mpg	dffit	cov.r	cook.d	hat	inf
Datsun 710	-0.001	0.001	0.000	-0.001	1.151	0.000	0.001	
Ferrari Dino	-0.021	0.022	0.019	-0.023	1.186	0.000	0.033	
Merc 280C	-0.146	0.150	0.144	0.222	1.212	0.003	0.106	
Merc 450SLC	-0.019	0.020	0.017	0.022	1.172	0.000	0.022	
Dodge Challenger	-0.076	0.106	0.031	0.443	1.166	0.012	0.153	
Maserati Bora	0.739	-0.961	-0.361	-2.965	0.029	0.528	0.140	*
Merc 450SL	-0.008	0.008	0.007	0.008	1.161	0.000	0.011	
AMC Javelin	1.194	-1.050	-1.327	2.283	0.729	0.297	0.397	*
Hornet 4 Drive	-0.245	0.231	0.286	0.362	1.332	0.008	0.199	
Volvo 142E	-0.091	0.094	0.073	-0.109	1.272	0.001	0.109	
Mazda RX4 Wag	-0.232	0.242	0.182	-0.297	1.389	0.005	0.210	
Merc 230	-0.236	0.220	0.279	0.335	1.391	0.006	0.219	
Fiat 128	-0.205	0.347	-0.205	-1.473	3.126	0.122	0.681	*
Ford Pantera L	-0.245	0.240	0.236	-0.263	1.416	0.004	0.216	*
Fiat X1-9	0.000	0.000	0.000	0.000	1.150	0.000	0.000	
Chrysler Imperial	0.000	0.000	0.000	0.000	1.150	0.000	0.000	
Hornet Sportabout	-0.100	0.101	0.101	0.125	1.221	0.001	0.082	
Merc 240D	-0.045	0.043	0.050	0.053	1.224	0.000	0.066	
Pontiac Firebird	0.000	0.000	0.000	0.000	1.151	0.000	0.001	
Cadillac Fleetwood	0.000	0.000	0.000	0.000	1.150	0.000	0.000	
Valiant	-0.111	0.113	0.110	0.145	1.218	0.001	0.086	
Lotus Europa	0.000	0.000	0.000	0.000	1.150	0.000	0.000	
Camaro Z28	-0.032	0.035	0.024	0.046	1.187	0.000	0.038	
Honda Civic	0.000	0.000	0.000	0.000	1.150	0.000	0.000	
Duster 360	0.062	0.000	-0.161	0.716	1.186	0.030	0.232	

Figura 15.15: estadísticas para la identificación de posibles casos influyentes.

```
29
30 # Ajustar modelo nulo
31 nulo <- glm(am ~ 1, family = binomial(link = "logit"), data = entrenamiento)
32
33 # Ajustar modelo completo
34 completo <- glm(am ~ ., family = binomial(link = "logit"), data = entrenamiento)
35
36 # Ajustar modelo con regresión escalonada
37 mejor <- step(nulo, scope = list(lower = nulo, upper = completo),
38               direction = "both", trace = 0)
39 cat("\n")
40 cat("Modelo con regresión escalonada\n")
41 cat("-----")
42 print(summary(mejor))
43
44 # Confirmar si es el modelo encontrado el que genera errores de
45 # convergencia.
46 mejor <- glm(formula = am ~ wt + hp, family = binomial(link = "logit"),
47               data = entrenamiento)
48
49 # Revisar, porque genera errores de convergencia
50 cat("Comparación con el modelo nulo\n")
```

```

51 cat("-----\n")
52 print(anova(nulo, mejor, test = "LRT"))
53
54 # Verificar multicolinealidad
55 vifs_completo <- vif(completo)
56 cat("\n")
57 cat("Multicolinealidad modelo completo\n")
58 cat("-----\n")
59 cat("VIF:\n")
60 print(vifs_completo[, 1])
61 cat("\n")
62 cat("VIF promedio:")
63 print(mean(vifs_completo[, 1]))
64
65 # Ver un paso adelante del modelo nulo
66 cat("\n")
67 cat("Primer paso\n")
68 cat("-----\n")
69 print(add1(nulo, scope = completo))
70
71 # Agregar la variable 'wt'
72 modelo_1 <- update(nulo, ". ~ . + wt")
73
74 # No generó problemas de convergencia
75 # Decartar las variables hp, qsec y carb
76 completo <- update(completo, ". ~ . -hp -qsec -carb")
77
78 # Ver un paso adelante
79 cat("\n")
80 cat("Segundo paso\n")
81 cat("-----\n")
82 print(add1(modelo_1, scope = completo))
83
84 # Agregar la variable 'mpg'.
85 modelo_2 <- update(modelo_1, ". ~ . + mpg")
86
87 # Decartar las variables drat y gear
88 completo <- update(completo, ". ~ . -disp -drat -gear")
89
90 # No genera errores de convergencia del modelo.
91 # Ver un paso adelante
92 cat("\n")
93 cat("Tercer paso\n")
94 cat("-----\n")
95 print(add1(modelo_2, scope = completo))
96
97 # Descartar todos los otros posibles predictores
98
99 # Comparar los modelos obtenidos
100 cat("\n")
101 cat("Likelihood Ratio Test para los modelos\n")
102 cat("-----\n")
103 print(anova(nulo, modelo_1, modelo_2, test = "LRT"))
104
105 # Verificar multicolinealidad
106 vifs_2 <- vif(modelo_2)
107 cat("\n")
108 cat("Verificación de colinealidad\n")
109 cat("-----\n")
110 cat("VIF:\n")

```

```

111 print(vifs_2)
112 cat("\n")
113 cat("VIF promedio:")
114 print(mean(vifs_2))
115
116 # Verificar linealidad con los predictores
117 datos_lin_w <- entrenamiento %>%
118   select(all_of(c("wt", "mpg"))) %>%
119   mutate(Logit = psych::logit(fitted(modelo_2)))
120 datos_lin_l <- pivot_longer(datos_lin_w, c(wt, mpg),
121                             names_to = "Predictor", values_to = "Valor")
122 p_1 <- ggscatter(datos_lin_l, x = "Logit", y = "Valor", ylab = "Valor del predictor
123 ")
124 p_1 <- p_1 + geom_smooth(method = lm, formula = y ~ x, se = TRUE)
125 p_1 <- p_1 + theme_pubr()
126 p_1 <- p_1 + facet_wrap(~ Predictor, scales = "free_y")
127 print(p_1)
128
129 # Verificar independencia de los residuos
130 cat("\n")
131 cat("Verificación de independencia de los residuos\n")
132 cat("-----\n")
133 print(durbinWatsonTest(modelo_2))
134
135 # Verificar normalidad de los residuos
136 p_2 <- autoplot(modelo_2, which = 2, label.colour = 'blue', label.repel = TRUE)
137 print(p_2)
138
139 # Revisar posibles casos influyentes
140 estad_inf <- influence.measures(modelo_2)
141 estad_inf$infmat <- round(estad_inf$infmat, 3)
142 cat("\n")
143 cat("Casos sospechosos de apalancamiento\n")
144 cat("-----\n")
145 print(estad_inf)

```

15.9 EJERCICIOS PROPUESTOS

1. Evalúa el clasificador conseguido con el script 15.3 en datos no vistos y compara el desempeño con los datos de entrenamiento. ¿Hay indicios de sobreajuste?.
2. Investiga cómo usar la función `roc.test()` del paquete `pROC` y verifica si la curva ROC generada por el modelo para los datos de entrenamiento es significativamente mejor que la generada para los datos de prueba.
3. Reconstruye el modelo de regresión logística conseguido con el script 15.3, pero ahora sin considerar los primeros casos influyentes. Evalúa la capacidad predictiva del modelo en los datos no vistos. ¿Cómo se compara con el modelo original?
4. Investiga cómo solicitar a la función `train()` del paquete `caret` que use alguna métrica distinta a la exactitud en su búsqueda de un modelo de regresión logística.
5. Hecho lo anterior, reconstruye el modelo de regresión logística conseguido con el script 15.3 usando validación cruzada dejando uno fuera buscando la mejor sensibilidad.
6. De igual forma, reconstruye el modelo de regresión logística conseguido con el script 15.3 usando validación cruzada de 5 pliegues buscando la ROC-AUC.
7. Investiga de qué se trata el método Recursive Feature Elimination implementado en el paquete `caret`. Bosqueja cómo se podría buscar un modelo como el obtenido en el script 15.3 utilizando este algoritmo.

15.10 BIBLIOGRAFÍA DEL CAPÍTULO

Ayala, J. (2020). *Minería de datos*.

Consultado el 23 de junio de 2021, desde <https://rpubs.com/JairoAyala/592802>

Cerda, J., Vera, C., & Rada, G. (2013). Odds ratio: aspectos teóricos y prácticos.

Revista médica de Chile, 141, 1329-1335.

DeLong, E. R., DeLong, D. M., & Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, 837-845.

Field, A., Miles, J., & Field, Z. (2012). *Discovering statistics using R*. SAGE Publications Ltd.

Glen, S. (2017). *Receiver Operating Characteristic (ROC) Curve: Definition, Example*. Consultado el 23 de junio de 2021, desde <https://www.statisticshowto.com/receiver-operating-characteristic-roc-curve/>

Zelada, C. (2017). *Evaluación de modelos de clasificación*.

Consultado el 23 de junio de 2021, desde <https://rpubs.com/chzelada/275494>