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 modelar cómo varía la media de una variable respuesta numérica Y a medida que cambian los valores de una o más variables predictoras $\mathbf{X} = (X_1, X_2, ..., X_k)$, cuando esta relación condicional sigue una distribución normal con varianza constante. Pero estas ideas se pueden **generalizar** considerando otras distribuciones de probabilidad.

Nelder y Wedderburn (1972) desarrollaron las bases de esta idea para distribuciones condicionales de la familia exponencial, como la binomial, la de Poisson, la gama y otras. De hecho, la regresión lineal que estudiamos en los capítulos anteriores puede verse como un caso especial de estos modelos, ya que la distribución normal también pertenece a la familia de distribuciones exponenciales.

En la **regresión lineal generalizada** (RLG), la respuesta depende de una función lineal de los predictores que puede tomar cualquier valor en $(-\infty, +\infty)$, llamado **predictor lineal**, como muestra la ecuación 15.1.

$$\eta(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k \tag{15.1}$$

En la regresión lineal, la relación entre el valor esperado de la variable respuesta (es decir, su media) y el predictor lineal es directa:

$$\mathbb{E}[y \mid \mathbf{x}] \equiv \mu_{y|\mathbf{x}} = \eta(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

Pero cuando la media condicional de la variable de salida tiene restricciones, como que solo pueda tomar valores positivos o en un intervalo determinado, es necesario usar una **función de enlace** invertible, que pueda convertir valores medios de la variable de salida desde y hacia la escala del predictor lineal, como se representa en la ecuación 15.2.

$$g(\mu_{y|\mathbf{x}}) = \eta(\mathbf{x}) \tag{15.2}$$

En el caso de la regresión lineal, es evidente que para el enlace se utiliza la función identidad: $I(\mu_{y|\mathbf{x}}) = \eta(\mathbf{x})$. Cambiando la función de enlace, entonces, podemos definir modelos de regresión que relacionan variables de salida que siguen diferentes distribuciones.

Los modelos de RLG se ajustan a los datos por el método de máxima verosimilitud, usando un algoritmo iterativo de **mínimos cuadrados ponderados**. Este procedimiento permite estimar los valores de los coeficientes de la regresión:

$$\hat{\eta}(\mathbf{x}) = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_k x_k \tag{15.3}$$

y, de esta forma, una estimación del valor medio para la variable de salida:

$$\hat{\mu}_{y|\mathbf{x}} = g^{-1}[\hat{\eta}(\mathbf{x})] \tag{15.4}$$

15.1 LA RELACIÓN LOGÍSTICA

Después de la regresión lineal, el modelo de RLG más relevante es la **regresión logística** (RLog) que usa como función de enlace la función **logit**, dada por la ecuación 15.5, cuya inversa es la **función logística estándar**, dada por la ecuación 15.6. La figura 15.1 presenta gráficamente estas funciones.

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) \qquad \text{con } p \in (0,1) \tag{15.5}$$

$$\text{logística}(z) = \text{logit}^{-1}(z) = \frac{1}{1+e^{-z}} \qquad \text{con } z \in (-\infty, +\infty) \tag{15.6}$$

Si nos fijamos en la gráfica de la función logística, podemos observar que el eje vertical describe una **transición de cero a uno** asociada al aumento de los valores en el eje horizontal. Esto resulta útil para representar **cómo cambia la probabilidad** de que algún evento ocurra en función del valor de una variable continua.

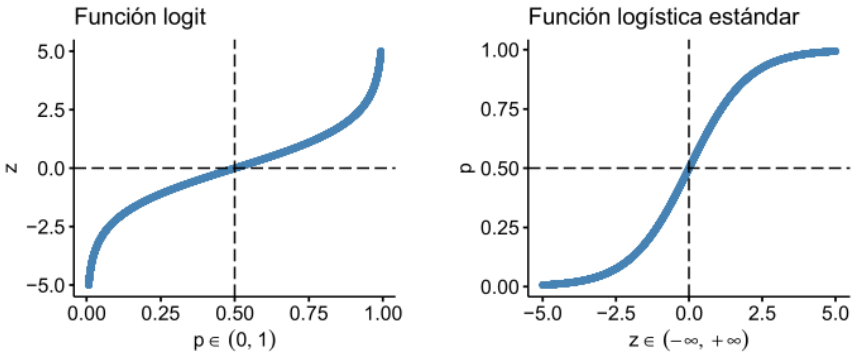


Figura 15.1: la función logit (izquierda) y, su inversa, la función logística (derecha).

De este modo, la regresión logística puede ser usada para modelar variables respuesta **binarias** que puedan tomar los valores 0 o 1, cuya media, en consecuencia, se encuentra en el intervalo $[0, 1]$ y que corresponde a la probabilidad de éxito: $\Pr(y = 1)$. Pero ahora, además, podemos mejorar la estimación de esta probabilidad considerando el valor observado para un conjunto de variables predictoras: $\Pr(y = 1 \mid \mathbf{x})$. La relación entre la media de esta probabilidad condicional ($\mu_{\Pr(y = 1 \mid \mathbf{x})}$) y los predictores queda definida por la ecuaciones:

$$\ln\left(\frac{\mu_{\Pr(y = 1 \mid \mathbf{x})}}{1 - \mu_{\Pr(y = 1 \mid \mathbf{x})}}\right) = \eta(\mathbf{x}) \tag{15.7}$$

$$\mu_{\Pr(y = 1 \mid \mathbf{x})} = \frac{1}{1 + e^{-\eta(\mathbf{x})}} \tag{15.8}$$

Si seleccionamos una división o **umbral** de probabilidad, podemos **predecir** la ocurrencia de un evento o, de forma equivalente, **clasificar** un caso en dos categorías posibles:

- Para valores menores que el umbral se predice que el evento "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 $\mu_{\Pr(y = 1 \mid \mathbf{x})} = 0{,}5$ como umbral, esto no es obligatorio ni siempre conveniente, como veremos más adelante. De esta forma, un modelo de RLog puede utilizarse como un **clasificador**.

15.2 EVALUACIÓN DE UN CLASIFICADOR

Una forma de evaluar modelos de clasificación, entre ellos los de RLog, es de acuerdo a la cantidad de errores cometidos (Tharwat, 2021). 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		
		1 (+)	**0** (−)	**Total**
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 clasificador corresponde a la proporción de observaciones correctamente clasificadas, dada por la ecuación 15.9.

$$\text{exactitud} = \frac{VP + VN}{n} \tag{15.9}$$

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

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

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

$$\text{sensibilidad} = \frac{VP}{VP + FN} \tag{15.11}$$

De manera análoga, la **especificidad** (*specificity*, ecuación 15.12) permite evaluar la aptitud del clasificador para asignar observaciones correctamente a la clase negativa.

$$\text{especificidad} = \frac{VN}{FP + VN} \tag{15.12}$$

La **precisión** (*precision*) o valor predictivo positivo (VPP, ecuación 15.13) indica cuán exacta es la asignación de elementos a la clase positiva (la proporción de instancias clasificadas como positivas que realmente lo son).

$$VPP = \frac{VP}{VP + FP} \tag{15.13}$$

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

$$VPN = \frac{VN}{FN + VN} \tag{15.14}$$

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 (Fawcett, 2006). Este gráfico permite evaluar visualmente la calidad del clasificador, puesto que mientras más se **aleje la curva de la diagonal**, mejor es su clasificación. Para ilustrar mejor la utilidad de este gráfico, la figura 15.2 muestra las curvas ROC para dos modelos diferentes, además de la diagonal (línea recta). Esta figura indica que el clasificador representado por la curva más oscura es mejor que el representado por la curva más clara, 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 clasificador perfecto que asigna correctamente la clase a todas las instancias exhibe un AUC = 1, mientras que uno que no discrimina, es decir, su desempeño no es mejor que el de una clasificación aleatoria, se asocia a un AUC = 0,5.

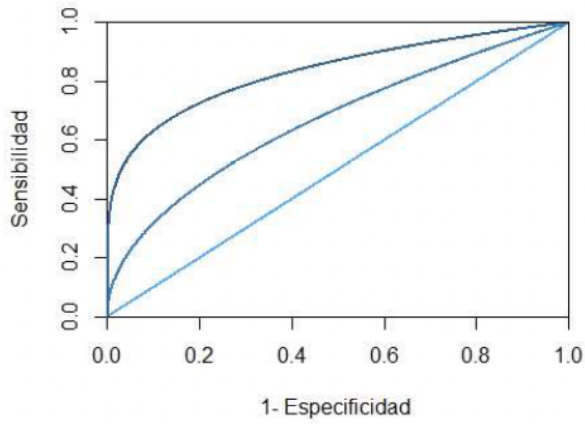


Figura 15.2: dos curvas ROC. Adaptado de Ayala (2020).

15.3 AJUSTE DE UN MODELO DE RLOG

Como se dijo anteriormente, el ajuste de un modelo de RLG se realiza mediante la resolución de un problema de optimización que busca minimizar las desviaciones entre las respuestas predichas y las observadas usando la función de verosimilitud. Si se tiene una muestra $D = \{(\mathbf{x}_j, y_j)\}$ con d observaciones, donde $j = \{1, 2, \ldots, d\}$, $y_j \in \{0, 1\}$ y $\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jn})$ 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 $B = (\beta_0, \beta_1, \ldots, \beta_n)$ es:

$$\begin{aligned}\mathcal{L}(B; D) &= \Pr(y_1, y_2, \ldots, y_d | X_1, X_2, \ldots, X_d; B) \\ &= \prod_{j=1}^{d} \Big[\Pr\big(y_j = 1 \mid \eta_B(\mathbf{x}_j)\big)^{y_j} \ \big[1 - \Pr\big(y_j = 1 \mid \eta_B(\mathbf{x}_j)\big)\big]^{1-y_j}\Big] \\ &= \prod_{j=1}^{d} \Big[\text{logística}\big(\eta_B(\mathbf{x}_j)\big)^{y_j} \ \big[1 - \text{logística}\big(\eta_B(\mathbf{x}_j)\big)\big]^{1-y_j}\Big]\end{aligned}$$

Por conveniencia numérica, se suele optimizar el logaritmo natural de la verosimilitud:

$$\ln \mathcal{L}(B; D) = \sum_{j=1}^{d} \Big[y_j \ \ln \text{logística}\big(\eta_B(\mathbf{x}_j)\big) + (1 - y_j) \ \ln \big[1 - \text{logística}\big(\eta_B(\mathbf{x}_j)\big)\big]\Big] \tag{15.15}$$

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 15.15, 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 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 15.16.

$$-2LL = -2 \ln \mathcal{L}(B; D) \tag{15.16}$$

Para comparar dos modelos de RLog M_1 y M_2 cuando el segundo es una extensión del primero, es decir M_2 contiene todos los predictores de M_1 más otros k predictores, se puede usar la diferencia de sus desviaciones,

$(-2LL_1) - (-2LL_2)$, que **sigue asintóticamente** (a medida que el tamaño de la muestra crece al infinito) una distribución χ^2 con k grados de libertad. 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 15.17, donde k corresponde a la cantidad de predictores en el modelo.

$$\text{AIC} = -2LL + 2k \tag{15.17}$$

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 15.18.

$$\text{BIC} = -2LL + 2k \ln n \tag{15.18}$$

15.4 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_respuesta> ~ <variable_predictora>`.
- `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 logit como enlace.

Siguiendo el ejemplo de los capítulos anteriores con el conjunto de datos `mtcars` (descrito en la tabla 13.1), el script 15.1 muestra la construcción de un modelo de RLog que predice el tipo de transmisión de un automóvil a partir de su peso.

La variable "tipo de transmisión" (`am`) es dicotómica, por lo que es ideal para este ejemplo. Si bien en el conjunto de datos viene etiquetada con los valores numéricos 0 para automóviles automáticos y 1 para los mecánicos, el script se encarga de convertirla en un factor con etiquetas más intuitivas (líneas 8–9).

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

```
1  library(caret)
2  library(dplyr)
3  library(ggpubr)
4  library(pROC)
5
6  # Cargar y filtrar los datos, teniendo cuidado de dejar "automático" como el
7  # 2do nivel de la variable "am" para que sea considerada como la clase positiva.
8  datos <- mtcars |> filter(wt > 2 & wt < 5) |>
9    mutate(am = factor(am, levels = c(1, 0), labels = c("manual", "automático")))
10
11 # Separar conjuntos de entrenamiento y prueba, fijando una semilla para asegurar
12 # la reproducibilidad de los resultados.
13 set.seed(101)
14 n <- nrow(datos)
15 i_muestra <- sample.int(n = n, size = floor(0.7 * n), replace = FALSE)
16 datos_ent <- datos[i_muestra, ]
17 datos_pru  <- datos[-i_muestra, ]
18
19 # Ajustar y mostrar modelo
20 modelo <- glm(am ~ wt, family = binomial(link = "logit"), data = datos_ent)
21 print(summary(modelo), signif.stars = FALSE)
22 cat("---\n")
23 print(anova(modelo, test = "LRT"), signif.stars = FALSE)
```

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

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -18.711      8.790  -2.129   0.0333
wt             6.003      2.741   2.191   0.0285

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 23.0348  on 16  degrees of freedom
Residual deviance:  9.9658  on 15  degrees of freedom
AIC: 13.966

Number of Fisher Scoring iterations: 6

---
Analysis of Deviance Table

Model: binomial, link: logit

Response: am

Terms added sequentially (first to last)


     Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                    16    23.0348
wt    1   13.069        15     9.9658 0.0003002
```

Figura 15.3: ajuste de un modelo de regresión logística.

Luego, siguiendo lo aprendido en los capítulos anteriores sobre construcción de modelos, se selecciona aleatoriamente un conjunto de entrenamiento, para construir el modelo, con el 70 % de las observaciones disponibles, dejando el restante 30 % para la validación (líneas 13–17). A continuación, las línea 20–21 del script muestran el uso de la función `glm()` para ajustar y mostrar en pantalla, respectivamente, el modelo deseado. Finalmente, la línea 23 muestra cómo comparar el modelo obtenido con el modelo nulo para determinar si es significativamente mejor aplicando la prueba LRT.

El resultado se muestra en la figura 15.3, donde podemos apreciar, en el resumen del modelo, que el peso del vehículo (`wt`) aporta significativamente al modelo ($z = 2{,}191$; $p = 0{,}029$) si consideramos un nivel de significación $\alpha = 0{,}05$, que la desviación para este modelo con un predictor (15 grados de libertad) se redujo a 9,966 del valor 23,035 exhibido por el modelo nulo, y que su valor para el criterio de información de Akaike es AIC = 13,966. En la segunda parte, vemos que la reducción en la desviación conseguida (13,069) es significativa ($\chi^2(15) = 13{,}069$; $p < 0{,}001$). Podemos concluir, entonces, que el modelo con un predictor presenta un mejor ajuste que el modelo nulo.

El script 15.2 muestra código R que permite evaluar la calidad predictiva del modelo ajustado en el conjunto de entrenamiento. La línea 31 muestra el uso de la función `roc(response, predictor, levels, direction` del paquete `pROC`, donde:

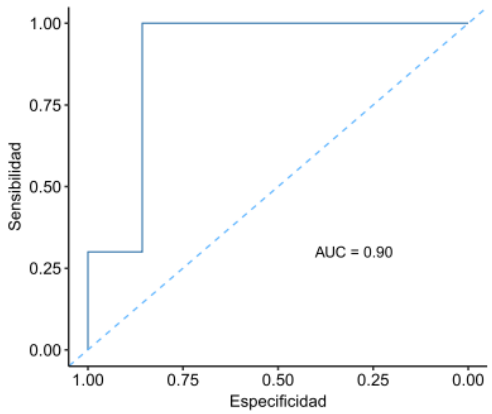
- `response`: factor con la clase observada (real o de referencia) para cada caso a considerar.
- `predictor`: vector con la puntuación que utiliza el modelo para predecir la clase de cada caso.
- `levels`: vector con dos strings que indican, respectivamente, la clase negativa y positiva en `response`.
- `direction`: indica si una mayor probabilidad de éxito (la clase positiva definida anteriormente) está asociada a valores más altos (`direction = ">"`) o más bajos (`direction = "<"`) del `predictor`.

La función `roc()` devuelve un objeto que representa la ROC que se genera al ir cambiando el umbral del valor del `predictor` usado para asignar la clase positiva a un caso. Existen otros argumentos optativos que permiten suavizar la curva, cambiar las escalas, entre otras cosas, que pueden revisarse en la documentación de R.

Script 15.2: (continuación del script 15.1) evaluación del modelo de RLog en los datos de entrenamiento.

```
25 #
26 # Evaluar el modelo con el conjunto de entrenamiento
27 #
28 probs_ent <- fitted(modelo)
29
30 # Graficar curva ROC, indicando AUC obtenido.
31 ROC_ent <- roc(datos_ent[["am"]], probs_ent,
32                levels = c("manual", "automático"), direction = "<")
33 g_ROC_ent <- ggroc(ROC_ent, color = "steelblue")
34 g_ROC_ent <- g_ROC_ent + geom_abline(intercept = 1, slope = 1,
35                                      colour = "steelblue1", linetype = "dashed")
36 g_ROC_ent <- g_ROC_ent + xlab("Especificidad") + ylab("Sensibilidad")
37 texto_ent <- sprintf("AUC = %.2f", ROC_ent[["auc"]])
38 g_ROC_ent <- g_ROC_ent + annotate("text", x = 0.3, y = 0.3, label = texto_ent)
39 g_ROC_ent <- g_ROC_ent + theme_pubr()
40 print(g_ROC_ent)
41
42 # Obtener las predicciones.
43 umbral <- 0.5
44 preds_ent <- sapply(probs_ent,
45                     function(p) ifelse(p >= umbral, "automático", "manual"))
46 preds_ent <- factor(preds_ent, levels = c("manual", "automático"))
47
48 # Obtener y mostrar estadísticas de clasificación en datos de entrenamiento.
49 mat_conf_ent <- confusionMatrix(preds_ent, datos_ent[["am"]],
50                                 positive = "automático")
51 cat("\n\nEvaluación de la calidad predictora (cjto. de entrenamiento):\n")
52 cat("-----------------------------------------------------------\n")
53 print(mat_conf_ent[["table"]])
54 cat("\n")
55 cat(sprintf("    Exactitud: %.3f\n", mat_conf_ent[["overall"]]["Accuracy"]))
56 cat(sprintf(" Sensibilidad: %.3f\n", mat_conf_ent[["byClass"]]["Sensitivity"]))
57 cat(sprintf("Especificidad: %.3f\n", mat_conf_ent[["byClass"]]["Specificity"]))
```

En la llamada a la función `roc()` utilizamos los valores ajustados por el modelo, recuperados en la línea 28. Recordemos que estos corresponden a las probabilidades de éxito para cada caso (ecuación 15.8). Las líneas 33–40 permiten obtener (y adornar) el gráfico de la curva, la que es mostrada al lado izquierdo de la figura 15.4. Podemos notar que esta se aleja bastante de la diagonal, por lo que al parecer se trata de un buen clasificador.




```
Evaluación del modelo (cjto. de entrenamiento):
-----------------------------------------------
            Reference
Prediction   manual automático
  manual          6          0
  automático      1         10

    Exactitud: 0.941
 Sensibilidad: 1.000
Especificidad: 0.857
```

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

En las líneas 43–46 del script se clasifica como un vehículo automático a todos los casos de entrenamiento cuya probabilidad de pertenecer a esta categoría sea igual o superior al tradicional umbral del 50 %. Con esta clasificación realizada, se puede obtener una matriz de confusión y calcular las métricas de evaluación del modelo. Esta tarea se facilita enormemente si se usa la función `confusionMatrix(data, reference, positive)` del paquete `caret`, donde `data` corresponde a las clases predichas, `reference` a las clases observadas y `positive` al nombre de la clase positiva. Las líneas 49–50 del script 15.2 utiliza esta función con la clasificación obtenida anteriormente, y las líneas 51–57 despliegan en pantalla algunas de las métricas de interés. El resultado de estas últimas se muestra en el lado derecho de la figura 15.4, donde podemos ver que el modelo tiene una exactitud de 94,1 %. La sensibilidad de 100 % y la especificidad de 85,70 % 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, esta evaluación puede estar sesgada y debemos evaluar el modelo con un conjunto de datos diferente al que usamos para su construcción. El script 15.3 realiza esta evaluación. En la línea 62, al indicar `type = "response"` en la función `predict()`, se obtienen las predicciones del modelo de la probabilidad de pertenecer a la clase positiva para cada caso de prueba. Luego, las líneas 65–74 obtienen y despliegan la ROC asociada a estas probabilidades. A continuación, las líneas 77–79 genera la clasificación de estos datos en las categorías de la variable de salida, la que es evaluada en las líneas 82–90.

El resultado del script puede verse en la figura 15.5, donde observamos una caída importante de la exactitud y la especificidad con respecto al rendimiento visto 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 el valor con los datos de entrenamiento (0,90) es menor al obtenido con los datos de prueba (0,92).

Script 15.3: (continuación del script 15.2) evaluación del modelo de RLog en los datos de prueba.

```
59 #
60 # Evaluar el modelo con el conjunto de prueba.
61 #
62 probs_pru <- predict(modelo, datos_pru, type = "response")
63
64 # Graficar curva ROC, indicando AUC obtenido.
65 ROC_pru <- roc(datos_pru[["am"]], probs_pru,
66                levels = c("manual", "automático"), direction = "<")
67 g_ROC_pru <- ggroc(ROC_pru, color = "steelblue")
68 g_ROC_pru <- g_ROC_pru + geom_abline(intercept = 1, slope = 1,
69                                      colour = "steelblue1", linetype = "dashed")
70 g_ROC_pru <- g_ROC_pru + xlab("Especificidad") + ylab("Sensibilidad")
71 texto_pru <- sprintf("AUC = %.2f", ROC_pru[["auc"]])
72 g_ROC_pru <- g_ROC_pru + annotate("text", x = 0.3, y = 0.3, label = texto_pru)
73 g_ROC_pru <- g_ROC_pru + theme_pubr()
74 print(g_ROC_pru)
75
76 # Obtener las predicciones (con el mismo umbral).
77 preds_pru <- sapply(probs_pru,
78                     function(p) ifelse(p >= umbral, "automático", "manual"))
79 preds_pru <- factor(preds_pru, levels = c("manual", "automático"))
80
81 # Obtener y mostrar estadísticas de clasificación en datos de prueba.
82 mat_conf_pru <- confusionMatrix(preds_pru, datos_pru[["am"]],
83                                 positive = "automático")
84 cat("\n\nEvaluación del modelo (cjto. de prueba):\n")
85 cat("-------------------------------------\n")
86 print(mat_conf_pru[["table"]])
87 cat("\n")
88 cat(sprintf("    Exactitud: %.3f\n", mat_conf_pru[["overall"]]["Accuracy"]))
89 cat(sprintf(" Sensibilidad: %.3f\n", mat_conf_pru[["byClass"]]["Sensitivity"]))
90 cat(sprintf("Especificidad: %.3f\n", mat_conf_pru[["byClass"]]["Specificity"]))
```