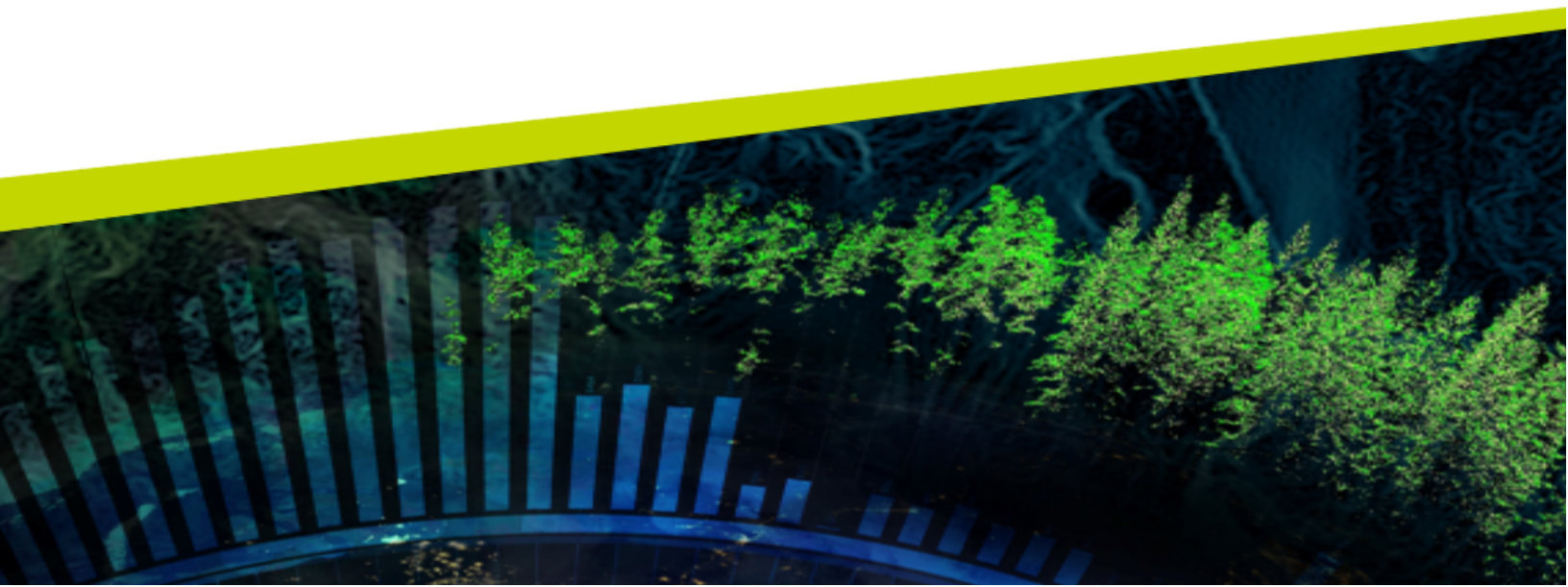
INFERENCIA Y MODELOS ESTADÍSTICOS

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 $\mathbf{X} = (X_1, X_2, ..., X_k)$, bajo la condición que 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.

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

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

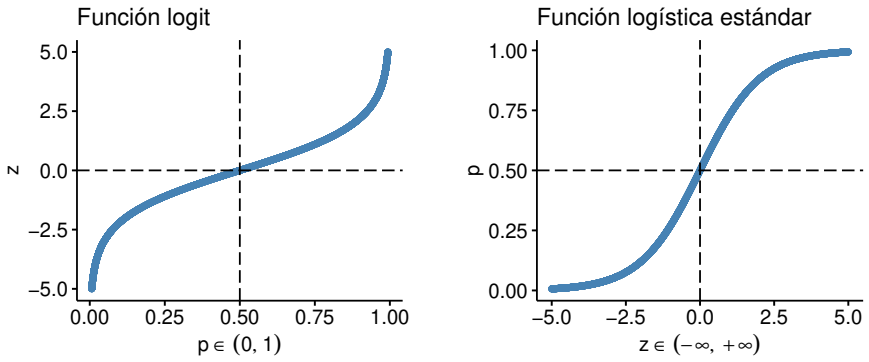


Figura 15.1: la función logit (izquierda) y, su inversa, la función logística (derecha).

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.

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 y los predictores queda definida por la ecuaciones:

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

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

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 $\Pr(y = 1 \mid \mathbf{x}) = 0{,}5$ 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 (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		
		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 determinar como la aptitud del clasificador para correctamente asignar observaciones 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 (Glen, 2017). Este gráfico permite evaluar 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. 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 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.

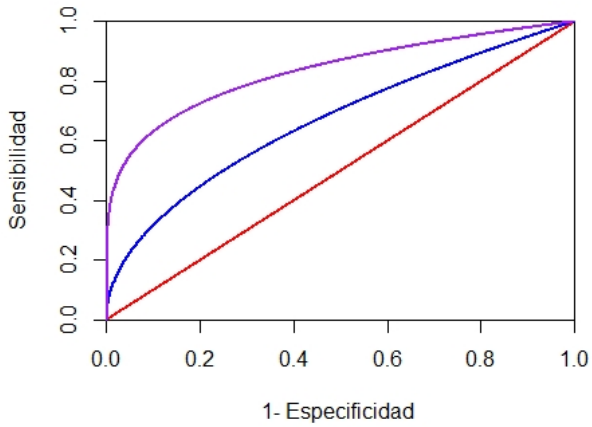


Figura 15.2: dos curvas ROC. Fuente: Ayala (2020).

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, aunque, por conveniencia, se suele optimizar el logaritmo natural de la 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:

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

Aplicando logaritmo natural se obtiene:

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

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, 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 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.

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

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 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 logit como enlace.

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, siguiendo el ejemplo de los capítulos anteriores con el conjunto de datos `mtcars` (descrito en la tabla 13.1).

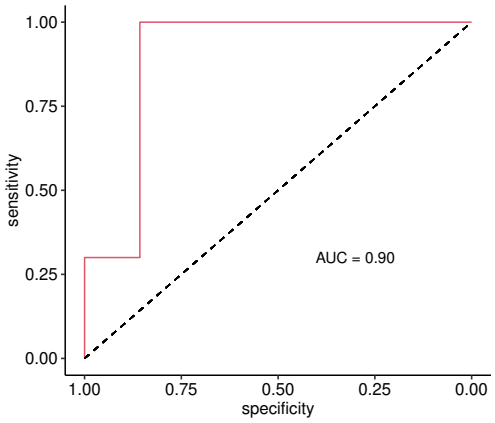
Las líneas 6–17 del script (incluyendo comentarios) ilustran la preparación de los datos para construir y evaluar apropiadamente el modelo buscado. Primero, se asegura que la variable de salida sea un factor dicotómico, puesto que la variable `am` viene originalmente con valores numéricos (0 = automática, 1 = manual). Luego, se selecciona aleatoriamente un conjunto de entrenamiento, para construir el modelo, con el 70 % de las observaciones disponibles. La línea 20–21 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 significativa en la desviación, que para este modelo con un predictor (18 grados de libertad) alcanza 12,725 y un criterio de información de Akaike AIC = 16,725.

Las líneas 23–53 del script 15.1 (incluyendo comentarios) evalúan la calidad predictiva del modelo ajustado en el conjunto de entrenamiento. La función `roc(response, predictor)` del paquete `pROC`, donde los argumentos corresponden, respectivamente, a las probabilidades observadas para cada caso 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 (izquierda). 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 probabilidades predichas, `reference` a las probabilidades observadas y `positive` al nombre de la clase positiva, genera la matriz de confusión y obtiene las medidas de evaluación de un clasificador descritas anteriormente, entre otras, como muestra la figura 15.4 (derecha). 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, debemos evaluar el modelo con un conjunto de datos diferente al que usamos para su construcción. Así, las líneas 55–84 (incluyendo comentarios) obtienen y despliegan la curva ROC y la matriz de confusión para el conjunto de prueba (figura 15.5), donde observamos un resultado con menor exactitud, sensibilidad y especificidad 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.

```
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 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(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
```

Figura 15.3: ajuste de un modelo de regresión logística.



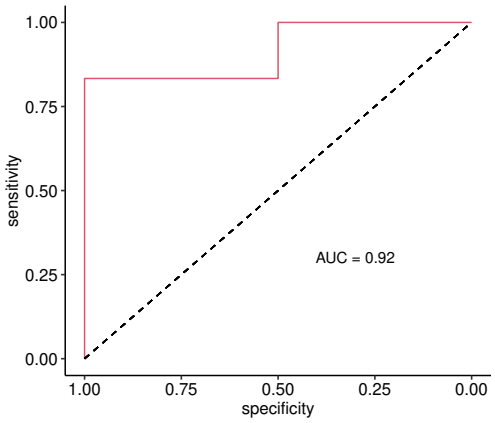

```
Evaluación del modelo (cjto. de entrenamiento):
------------------------------------------------
            Reference
Prediction   manual automático
  manual          6          0
  automático      1         10

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

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

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). Podríamos determinar si estos valores son o no significativamente diferentes usando la función `roc.test()` del mismo paquete `pROC` considerando, por ejemplo, el método de DeLong y col. (1988).

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

```r
library(caret)
library(dplyr)
library(ggpubr)
library(pROC)

# Cargar y filtrar los datos, teniendo cuidado de dejar
# "automático" como 2do nivel de la variable "am" para que
# sea considerada como la clase positiva.
datos <- mtcars |> filter(wt > 2 & wt < 5) |>
  mutate(am = factor(am, levels = c(1, 0), labels = c("manual", "automático")))

# Separar conjuntos de entrenamiento y prueba.
set.seed(101)
n <- nrow(datos)
i_muestra <- sample.int(n = n, size = floor(0.7 * n), replace = FALSE)
```




```
Evaluación del modelo (cjto. de prueba):
----------------------------------------
            Reference
Prediction   manual automático
  manual          1          1
  automático      1          5

    Exactitud: 0.750
 Sensibilidad: 0.833
Especificidad: 0.500
```

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

```r
datos_ent <- datos[i_muestra, ]
datos_pru  <- datos[-i_muestra, ]

# Ajustar modelo.
modelo <- glm(am ~ wt, family = binomial(link = "logit"), data = datos_ent)
print(summary(modelo))

#
# Evaluar el modelo con el conjunto de entrenamiento.
#
probs_ent <- fitted(modelo)

# Graficar curva ROC, indicando AUC obtenido.
ROC_ent <- roc(datos_ent[["am"]], probs_ent)
texto_ent <- sprintf("AUC = %.2f", ROC_ent[["auc"]])
g_roc_ent <- ggroc(ROC_ent, color = 2)
g_roc_ent <- g_roc_ent + geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1),
                                      linetype = "dashed")
g_roc_ent <- g_roc_ent + annotate("text", x = 0.3, y = 0.3, label = texto_ent)
g_roc_ent <- g_roc_ent + theme_pubr()
print(g_roc_ent)

# Obtener las predicciones.
umbral <- 0.5
preds_ent <- sapply(probs_ent,
                    function(p) ifelse(p >= umbral, "automático", "manual"))
preds_ent <- factor(preds_ent, levels = levels(datos[["am"]]))

# Obtener y mostrar estadísticas de clasificación en datos de entrenamiento.
mat_conf_ent <- confusionMatrix(preds_ent, datos_ent[["am"]],
                                positive = "automático")
cat("\n\nEvaluación del modelo (cjto. de entrenamiento):\n")
cat("-----------------------------------------------\n")
print(mat_conf_ent[["table"]])
cat("\n")
cat(sprintf("    Exactitud: %.3f\n", mat_conf_ent[["overall"]]["Accuracy"]))
cat(sprintf(" Sensibilidad: %.3f\n", mat_conf_ent[["byClass"]]["Sensitivity"]))
cat(sprintf("Especificidad: %.3f\n", mat_conf_ent[["byClass"]]["Specificity"]))

#
# Evaluar el modelo con el conjunto de prueba.
#
```

```
probs_pru <- predict(modelo, datos_pru, type = "response")

# Graficar curva ROC, indicando AUC obtenido.
ROC_pru <- roc(datos_pru[["am"]], probs_pru)
texto_pru <- sprintf("AUC = %.2f", ROC_pru[["auc"]])
g_roc_pru <- ggroc(ROC_pru, color = 2)
g_roc_pru <- g_roc_pru + geom_segment(aes(x = 1, xend = 0, y = 0, yend = 1),
                                      linetype = "dashed")
g_roc_pru <- g_roc_pru + annotate("text", x = 0.3, y = 0.3, label = texto_pru)
g_roc_pru <- g_roc_pru + theme_pubr()
print(g_roc_pru)

# Obtener las predicciones (con el mismo umbral).
preds_pru <- sapply(probs_pru,
                    function(p) ifelse(p >= umbral, "automático", "manual"))
preds_pru <- factor(preds_pru, levels = levels(datos[["am"]]))

# Obtener y mostrar estadísticas de clasificación en datos de prueba.
mat_conf_pru <- confusionMatrix(preds_pru, datos_pru[["am"]],
                                positive = "automático")
cat("\n\nEvaluación del modelo (cjto. de prueba):\n")
cat("-------------------------------------\n")
print(mat_conf_pru[["table"]])
cat("\n")
cat(sprintf("    Exactitud: %.3f\n", mat_conf_pru[["overall"]]["Accuracy"]))
cat(sprintf(" Sensibilidad: %.3f\n", mat_conf_pru[["byClass"]]["Sensitivity"]))
cat(sprintf("Especificidad: %.3f\n", mat_conf_pru[["byClass"]]["Specificity"]))
```

En capítulos anteriores conocimos la validación cruzada como herramienta para conseguir una estimación más confiable y robusta del rendimiento 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 4 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 15.1, seleccionando aleatoriamente un conjunto de entrenamiento y otro de prueba, cuatro veces.

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 variables 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 (88,7 %) y desviación estándar (13,1 %) 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.

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

```
library(caret)
library(data.table)
library(dplyr)

# Cargar y filtrar los datos, teniendo cuidado de dejar
# "automático" como 2do nivel de la variable "am" para que
# sea considerada como la clase positiva.
datos <- mtcars |> filter(wt > 2 & wt < 5) |>
  mutate(am = factor(am, levels = c(1, 0), labels = c("manual", "automático")))

```

```
Modelo RLog :
------------

Call:
NULL

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 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(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


Evaluación del modelo (cjto. de entrenamiento):
-----------------------------------------------
            Reference
Prediction   manual automático
  manual          6          1
  automático      1          9

    Exactitud: 0.882
 Sensibilidad: 0.900
Especificidad: 0.857


Detalle por pliegue:
--------------------
 Accuracy Resample
     1.00    Fold1
     0.80    Fold2
     0.75    Fold3
     1.00    Fold4
--------------------
    0.887     Mean
    0.131       SD


Evaluación del modelo (cjto. de prueba):
----------------------------------------
            Reference
Prediction   manual automático
  manual          1          1
  automático      1          5

    Exactitud: 0.750
 Sensibilidad: 0.833
Especificidad: 0.500
```

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

```r
# Separar conjuntos de entrenamiento y prueba.
set.seed(101)
n <- nrow(datos)
i_muestra <- sample.int(n = n, size = floor(0.7 * n), replace = FALSE)
datos_ent <- datos[i_muestra, ]
datos_pru  <- datos[-i_muestra, ]

# Ajustar modelo usando validación cruzada de 4 pliegues.
modelo_ent <- train(am ~ wt, data = datos_ent, method = "glm",
                    family = binomial(link = "logit"),
                    trControl = trainControl(method = "cv", number = 4,
                                             savePredictions = TRUE))
modelo <- modelo_ent[["finalModel"]]

cat("Modelo RLog :\n")
cat("------------\n")
print(summary(modelo))

# Obtener y mostrar estadísticas de clasificación en datos de entrenamiento.
mat_conf_ent <- confusionMatrix(modelo_ent[["pred"]][["pred"]],
                                modelo_ent[["pred"]][["obs"]],
                                positive = "automático")

cat("\nEvaluación del modelo (cjto. de entrenamiento):\n")
cat("-----------------------------------------------\n")
print(mat_conf_ent[["table"]])
cat("\n")
cat(sprintf("    Exactitud: %.3f\n", mat_conf_ent[["overall"]]["Accuracy"]))
cat(sprintf(" Sensibilidad: %.3f\n", mat_conf_ent[["byClass"]]["Sensitivity"]))
```

```
cat(sprintf("Especificidad: %.3f\n", mat_conf_ent[["byClass"]]["Specificity"]))

cat("\n\nDetalle por pliegue:\n")
cat("--------------------\n")
resumen <- data.table(modelo_ent[["resample"]][, c(1, 3)])
resumen <- rbind(resumen, list(modelo_ent[["results"]][[2]], "Mean"))
resumen <- rbind(resumen, list(modelo_ent[["results"]][[4]], "SD"))
print(resumen[1:4, ], row.names = FALSE)
cat("--------------------\n")
print(resumen[5:6, ], row.names = FALSE, col.names = "none", digits = 3)

# Obtener las predicciones en los datos de prueba.
umbral <- 0.5
probs <- predict(modelo, datos_pru, type = "response")
preds <- ifelse(probs >= umbral, "automático", "manual")
preds <- factor(preds, levels = levels(datos[["am"]]))

# Obtener y mostrar estadísticas de clasificación en datos de entrenamiento.
mat_conf_pru <- confusionMatrix(preds, datos_pru[["am"]], positive = "automático")

cat("\n\nEvaluación del modelo (cjto. de prueba):\n")
cat("---------------------------------------\n")
print(mat_conf_pru[["table"]])
cat("\n")
cat(sprintf("    Exactitud: %.3f\n", mat_conf_pru[["overall"]]["Accuracy"]))
cat(sprintf(" Sensibilidad: %.3f\n", mat_conf_pru[["byClass"]]["Sensitivity"]))
cat(sprintf("Especificidad: %.3f\n", mat_conf_pru[["byClass"]]["Specificity"]))
```

15.5 MÚLTIPLES DE PREDICTORES

Cuando tenemos múltiples predictores potenciales, debemos decidir cuáles de ellos incorporar en el modelo de RLog. 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, usando para ello las mismas funciones de R descritas en el capítulo 14.

En páginas previas ajustamos un modelo de RLog 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 22–23 del script 15.3 llevan a cabo esta tarea usando regresión escalonada, obteniéndose como resultado el modelo presentado en la figura 15.7, en donde podemos ver que, además del peso (`wt`), el mejor modelo encontrado considera además la potencia del motor (`hp`) como predictor.

Script 15.3: búsqueda de un modelo de RLog usando regresión escalonada.

```
library(ggpubr)
library(dplyr)

# Cargar y filtrar los datos (solo predictores numéricos).
datos <- mtcars |> filter(wt > 2 & wt < 5) |>
  select(-c("cyl", "vs", "gear", "carb")) |>
  mutate(am = factor(am, levels = c(1, 0), labels = c("manual", "automático")))

# Separar conjuntos de entrenamiento y prueba.
set.seed(101)
n <- nrow(datos)
i_muestra <- sample.int(n = n, size = floor(0.7 * n), replace = FALSE)
datos_ent <- datos[i_muestra, ]
```

```
Modelo RLog conseguido con regresión escalonada:
------------------------------------------------

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

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.730e+02  6.464e+05  -0.001        1
wt           1.409e+02  2.437e+05   0.001        1
hp          -4.994e-01  1.960e+03   0.000        1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2.3035e+01  on 16  degrees of freedom
Residual deviance: 4.6315e-10  on 14  degrees of freedom
AIC: 6

Number of Fisher Scoring iterations: 25

Warning messages:
1: glm.fit: fitted probabilities numerically 0 or 1 occurred
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
3: glm.fit: algorithm did not converge
4: glm.fit: fitted probabilities numerically 0 or 1 occurred
5: glm.fit: fitted probabilities numerically 0 or 1 occurred
6: glm.fit: fitted probabilities numerically 0 or 1 occurred
7: glm.fit: fitted probabilities numerically 0 or 1 occurred
8: glm.fit: fitted probabilities numerically 0 or 1 occurred
9: glm.fit: fitted probabilities numerically 0 or 1 occurred
10: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

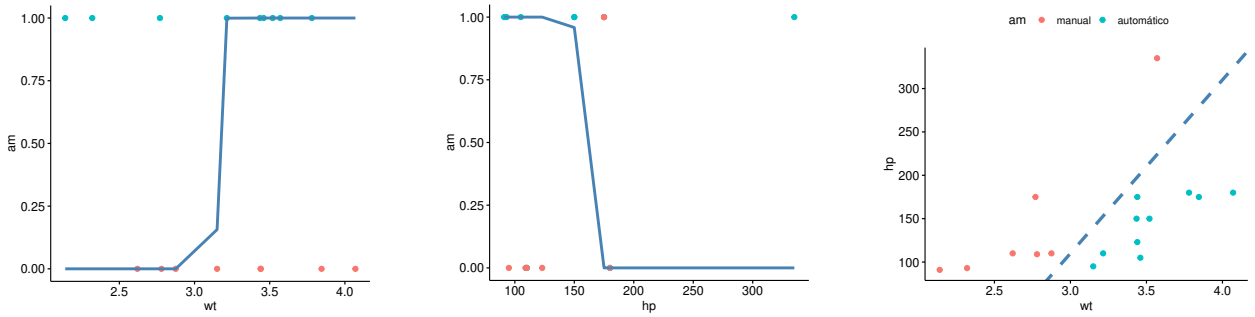
Figura 15.7: salida obtenida al buscar un modelo de RLog con regresión escalonada.

```r
datos_pru  <- datos[-i_muestra, ]

# Modelos inicial y máximo.
nulo <- glm(am ~ 1, family = binomial(link = "logit"), data = datos_ent)
maxi <- glm(am ~ ., family = binomial(link = "logit"), data = datos_ent)

# Ajustar modelo con regresión paso a paso escalonada.
modelo <- step(nulo, scope = list(upper = maxi),
               direction = "both", trace = FALSE)

cat("Modelo RLog conseguido con regresión escalonada:\n")
cat("------------------------------------------------\n")
print(summary(modelo))
```

Sin embargo, la figura 15.7 también podemos ver que aparecen varios mensajes de advertencia indicando que se encontraron problemas de convergencia al realizar la búsqueda de este modelo. Efectivamente, si nos fijamos en la evaluación de las estimaciones de los coeficientes, vemos que ¡estos tienen valor p igual a 1! Una clara indicación de que el modelo tiene problemas.

En un comienzo no es fácil determinar la raíz de los problemas. Si se grafican cómo serían los modelos de RLog asociados a cada predictor suponiendo constante el otro, obtenemos los gráficos (a) y (b) mostrados en la figura 15.8. Nada extraño es visible en ellos.

Sin embargo, cuando vemos cómo interactúan estas variables en el gráfico (c), podemos darnos cuentas que una simple recta puede hacer una **separación perfecta** de los vehículos con transmisión manual de



(a) `am ~ wt` (b) `am ~ hp` (c) `wt` vs. `hp`

Figura 15.8: gráficos de las predicciones parciales de los predictores peso (a) y potencia (b) y una visión de las clases en el gráfico de dispersión formado por estas variables (c).

los que tienen transmisión automática. Es decir, ¡no necesitamos un clasificador logístico! El algoritmo de optimización iterativo descrito en la sección 15.3 no puede converger a coeficientes estables este caso.

El script 15.4 muestra una regresión paso a paso hacia adelante, de manera que podamos ir evitando caer en este problema de separación perfecta. Primero se construyen los modelos nulo y completo para ser usados, respectivamente, como modelo inicial y límite superior de la búsqueda. Notemos que el ajuste del modelo completo genera la primera advertencia de problemas de convergencia, como se aprecia en la figura 15.9 muestra la salida que genera esta búsqueda en pantalla.

Comenzando con el modelo nulo, en el primer paso (columna izquierda) se encuentra que la mayor reducción en AIC se obtiene agregando el peso del vehículo como predictor. La línea 29 del script actualiza el modelo en concordancia. En el segundo paso (columna central), vemos que aparecen varias advertencias sobre problemas de convergencia. Vemos que la función `add1()` reporta desviación nula agregamos como predictores las variables `hp` o `qsec`. Ya sabíamos de la primera, ahora sabemos que tampoco podemos usar la segunda. Además, debemos observar que la única reducción del AIC se consigue agregando el rendimiento del vehículo (`mpg`) como predictor. La línea 37 del script 15.4 realiza la actualización del modelo de acuerdo a este resultado. Al revisar el siguiente paso (columna derecha de la figura 15.9), que genera más advertencias de problemas de convergencia, vemos que la mejor opción es detener la búsqueda y quedarnos con el modelo que incluye dos predictores.

La figura 15.10 presenta la salida a pantalla que genera la segunda parte del script 15.4. Vemos que, tanto al evaluar los coeficientes del modelo obtenido como al comparar el este modelo con los precedentes, pareciera que la inclusión del predictor `mpg` no hace un aporte importante al ajuste del modelo, y lo mejor sería evaluar su eliminación. Sin embargo, con lo pequeña que es esta muestra, no es tan aconsejable tomar decisiones cuando la evidencia está cerca del borde.

Script 15.4: búsqueda de un modelo de RLog usando regresión paso a paso hacia adelante.

```
library(ggpubr)
library(dplyr)

# Imprimir mensajes de advertencia a medida que ocurre.
opt <- options(warn = 1, width = 26)

# Cargar y filtrar los datos (solo predictores numéricos).
datos <- mtcars |> filter(wt > 2 & wt < 5) |>
  select(-c("cyl", "vs", "gear", "carb")) |>
  mutate(am = factor(am, levels = c(1, 0), labels = c("manual", "automático")))

# Separar conjuntos de entrenamiento y prueba.
set.seed(101)
n <- nrow(datos)
i_muestra <- sample.int(n = n, size = floor(0.7 * n), replace = FALSE)
```

```
Warning: glm.fit: fitted
probabilities numerically
 0 or 1 occurred
Paso 1:
------
Single term additions

Model:
am ~ 1
       Df Deviance    AIC
<none>     23.0348 25.035
mpg     1  18.9591 22.959
disp    1  15.1710 19.171
hp      1  23.0306 27.031
drat    1  12.5165 16.517
wt      1   9.9658 13.966
qsec    1  18.4951 22.495
```

```
Paso 2:
------
Warning: glm.fit: fitted
probabilities numerically
Warning: glm.fit: algorithm
did not converge
Warning: glm.fit: fitted
probabilities numerically
Single term additions

Model:
am ~ wt
       Df Deviance    AIC
<none>      9.9658 13.966
mpg     1   6.2682 12.268
disp    1   9.7704 15.770
hp      1   0.0000  6.000
drat    1   9.2023 15.202
qsec    1   0.0000  6.000
```

```
Paso 3:
------
Warning: glm.fit: fitted
probabilities numerically
Warning: glm.fit: algorithm
did not converge
Warning: glm.fit: fitted
probabilities numerically
Warning: glm.fit: algorithm
did not converge
Warning: glm.fit: fitted
probabilities numerically
Single term additions

Model:
am ~ wt + mpg
       Df Deviance    AIC
<none>      6.2682 12.268
disp    1   6.0578 14.058
hp      1   0.0000  8.000
drat    1   0.0000  8.000
qsec    1   0.0000  8.000
```

Figura 15.9: salida obtenida al construir un modelo de RLog manualmente con regresión hacia adelante.

```r
datos_ent <- datos[i_muestra, ]
datos_pru  <- datos[-i_muestra, ]

# Definir modelos inicial y máximo.
nulo <- glm(am ~ 1, family = binomial(link = "logit"), data = datos_ent)
maxi <- glm(am ~ ., family = binomial(link = "logit"), data = datos_ent)

# Revisar un paso hacia adelante.
cat("\nPaso 1:\n")
cat("------\n")
print(add1(nulo, scope = maxi))

# Actualizar el modelo.
modelo1 <- update(nulo, . ~ . + wt)

# Revisar un paso hacia adelante.
cat("\nPaso 2:\n")
cat("------\n")
print(add1(modelo1, scope = maxi))

# Actualizar el modelo.
modelo2 <- update(modelo1, . ~ . + mpg)

# Revisar un paso hacia adelante.
cat("\nPaso 3:\n")
cat("------\n")
print(add1(modelo2, scope = maxi))

# Mostrar el modelo obtenido.
cat("\nModelo RLog conseguido con regresión hacia adelante:\n")
cat("-------------------------------------------------\n")
print(summary(modelo2))

# Comparar los modelos generados.
```

```
Modelo RLog conseguido con regresión hacia adelante:
-----------------------------------------------

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

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -66.4664    39.3397  -1.690   0.0911 .
wt           14.4710     8.0062   1.807   0.0707 .
mpg           1.0849     0.8049   1.348   0.1777
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 23.0348  on 16  degrees of freedom
Residual deviance:  6.2682  on 14  degrees of freedom
AIC: 12.268

Number of Fisher Scoring iterations: 7

Comparación de los modelos considerados:
---------------------------------------
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        16    23.0348
2        15     9.9658  1  13.0690 0.0003002 ***
3        14     6.2682  1   3.6976 0.0544915 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figura 15.10: modelo de RLog que resulta de aplicar manualmente regresión hacia adelante y su comparación con otros modelos evaluados durante la búsqueda.

```r
cat("Comparación de los modelos considerados:\n")
cat("---------------------------------------\n")
print(anova(nulo, modelo1, modelo2, test = "LRT"))

# Reestabler opción para warnings
options(warn = opt[[1]], width = opt[[2]])
```

15.6 CONFIABILIDAD DE UN MODELO DE RLOG

Aprovechando las dificultades encontradas con el ejemplo, es importante recordar que no basta con realizar una búsqueda automática de predictores y evaluar el desempeño de un clasificador, sino que también es esencial 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.

Si bien son menos condiciones que las vistas para la regresión lineal, debemos además agregar situaciones en que puede ocurrir que el método de optimización no converja:

3. Multicolinealidad entre los predictores, que en este caso se evalúa y aborda del mismo modo que para RLM (por ejemplo, mediante el factor de inflación de la varianza o la tolerancia).
4. Información incompleta, que se produce cuando no contamos con observaciones suficientes para todas las posibles combinaciones de predictores, en especial para algún nivel de una variable categórica.
5. Separación perfecta, que ocurre cuando no hay superposición entre las clases (es decir, como vimos, cuando los predictores separan ambas clases completamente).

Y finalmente, también debemos descartar la presencia de casos problemáticos:

6. Las estimaciones de los coeficientes del modelo no están dominadas por casos influyentes.

Para verificar la condición 1 podemos usar los gráficos de residuos vistos para la RLM, provistos por la función `residualPlots()` del paquete `car`, aunque el gráfico de residuos versus los valores predichos pierde sentido en la RLog (y en los modelos de RLG), por lo que agregamos el argumento `fitted = FALSE`. El mismo paquete también permite revisar linealidad usando gráficos de los **residuos parciales**, es decir, cuánto contribuye a los residuos cada uno de los predictores, por medio de la función `crPlots(modelo)`. Estos tipos de gráficos se muestran, respectivamente, en la parte superior e inferior de la figura 15.11.

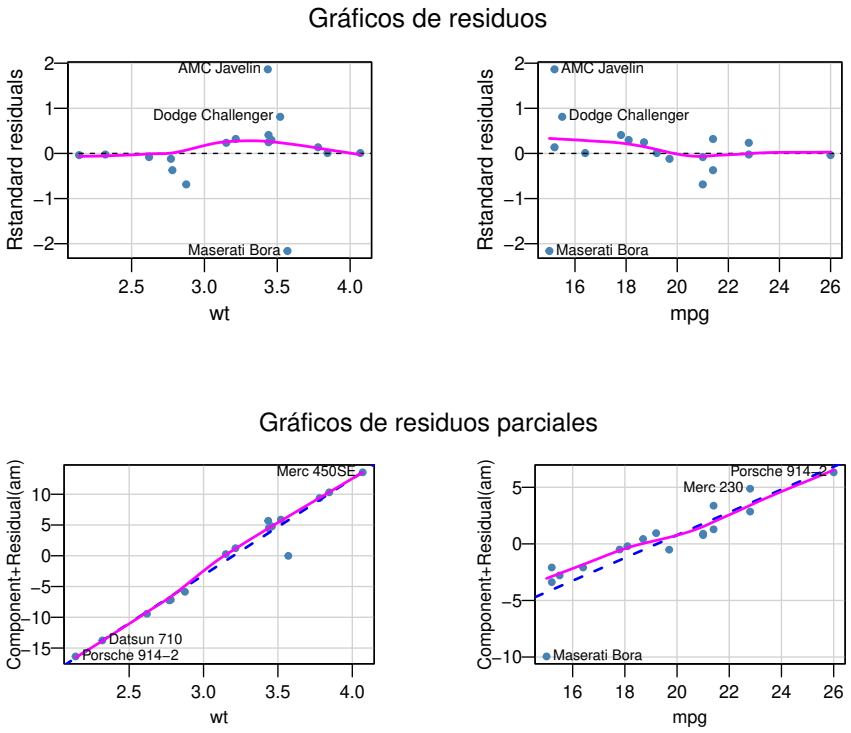


Figura 15.11: gráficos de residuos y gráficos de residuos parciales para evaluar la condición de linealidad en el modelo de RLog ejemplo.

En ambos gráficos, observamos que el supuesto de linealidad para cumplirse relativamente bien. De hecho, el resultado de las pruebas de curvatura que también realiza y reporta la función `residualPlots()` confirman esta idea:

```
    Test stat Pr(>|Test stat|)
wt     1.3077           0.2528
mpg    0.3709           0.5425
```

Pasando a la segunda condición, nuevamente podemos utilizar lo hecho para RLM y aplicar la prueba de Durbin-Watson implementada en la función `durbinWatsonTest` del paquete `car`, obteniendo:

```
 lag Autocorrelation D-W Statistic p-value
   1      -0.1794747      2.339983   0.532
 Alternative hypothesis: rho != 0
```

Con esto descartamos que exista evidencia para sospechar que no se esté cumpliendo la condición de independencia de los residuos.

Pasemos entonces a revisar que no existe no exista multicolinealidad fuerte, utilizando la ya conocida función `vif(modelo)` para obtener los factores de inflación de la varianza, que nos indica:

```
      wt      mpg
3.846939 3.846939
```

Vemos que reporta valores iguales ($VIF = 3{,}847$) 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 más umbral preocupante fijado en 5 ni el umbral crítico de 10. Si bien se mantienen estos umbrales, debemos notar que los valores de *VIF* provienen de modelos lineales auxiliares, y no directamente del modelo de regresión logística, por lo que debemos ser especialmente cuidadosos.

La siguiente condición es problemática para construir cualquier modelo. Recordemos que cuando estudiamos la RLM, se dijo que, si bien existe un criterio único del tamaño de la muestra requerido, se ha recomendado contar con 10 o 15 observaciones por cada predictor numérico y nivel de las variables categóricas. Esto no suele ser simple, puesto que, por ejemplo, la variable asociada al número de carburadores presentes (`carb`) tiene 4 niveles, por lo que si se considera este potencial predictor, se necesitarían más de 50 datos distribuidos más o menos balanceadamente en estos niveles. Además, habría que considerar que estas observaciones, en cada nivel, cubriera el rango de valores de cada predictor numérico. En el ejemplo, tenemos dos predictores numéricos y, si bien los casos en la muestra se distribuyen bien a lo largo de sus valores, solo tenemos 17 observaciones, muy pocas como para conseguir un modelo confiable.

Por último, queda revisar la condición relacionada a casos influyentes. Cuando revisamos los residuos parciales respecto de la variable `car` (gráfico inferior derecho de la figura 15.11), debimos notar que la recta ajustada (línea punteada en color azul-acero) esta algo desviada de la curva de ajuste local de los datos (línea sólida en tono rojizo). Esto parece deberse por la influencia de un caso muy alejado hacia los valores negativos, correspondiente al (ya sospechoso de siempre) modelo Maserati Bora. Sin embargo, esta influencia no se observa en los gráficos de residuos, tal vez moderada por la presencia de otro caso atípico, pero positivo, correspondiente al modelo AMC Javelin.

Como con los diagnósticos anteriores, aquí también podemos aplicar las herramientas usadas para RLM, como el gráfico burbuja con tres medidas de influencia para el modelo que se obtiene con la función `influencePlot(modelo)` del paquete `car`, que para el ejemplo se presenta en la figura 15.12.

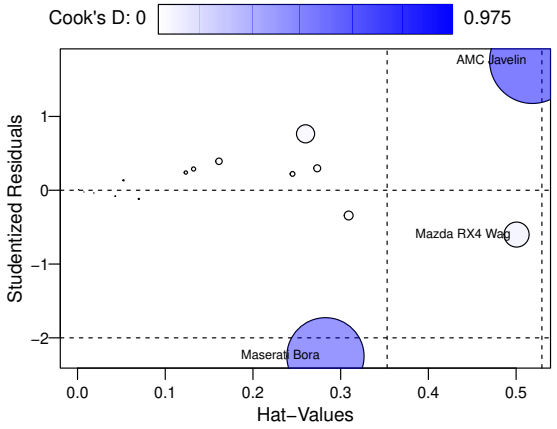


Figura 15.12: gráfico de influencia para el modelo de RLog del ejemplo.

Recordando que esta función también reporta los estadísticos para casos notorios, nos fijamos en lo que muestra en pantalla:

```
                 StudRes       Hat      CookD
AMC Javelin    1.7553843 0.5185520 0.97532006
Mazda RX4 Wag -0.6005575 0.5005603 0.08357138
Maserati Bora -2.2497803 0.2826453 0.79483991
```

Vemos entonces que hay tres observaciones que sobrepasan dos veces el apalancamiento promedio (0,353 aproximadamente para el modelo de RLog del ejemplo) y que dos de ellos también presentan distancias de Cook altas (sobre el umbral $4/17 = 0{,}235$) pero que no sobrepasan el valor crítico de 1 (aunque el modelo AMC Javelin está muy cerca de hacerlo). El modelo Maserati Bora también constituye claramente un valor atípico.

Como hemos dicho anteriormente, correspondería hacer un análisis riguroso de los datos para decidir si eliminar estas observaciones problemáticas sin violar los principios de la ética profesional de un analista de datos. Si así se decidiera, se debe reconstruir (rehacer la búsqueda de predictores y coeficientes) y reevaluar el modelo que se consiga sin datos influyentes. Y si vuelven a aparecer otros casos con demasiada influencia, se debe iterar nuevamente en la búsqueda de un modelo confiable. En este ejemplo, el problema puede deberse al tamaño insuficiente de la muestra de observaciones, por lo que si el estudio fuera nuestro, los esfuerzos deberían concentrarse en extender este conjunto.

15.7 EJERCICIOS PROPUESTOS

15.1 Evalúa la calidad predictiva del clasificador mostrado en la figura 15.10 en datos no vistos y compara el desempeño con los datos de entrenamiento. ¿Hay indicios de sobreajuste?.

15.2 Evalúa la calidad predictiva del clasificador mostrado en la figura 15.10 usando cinco repeticiones de validación cruzada de 4 pliegues con los datos de entrenamiento y luego en el conjunto de prueba. ¿Hay indicios de sobreajuste?.

15.3 Evalúa la calidad predictiva del clasificador mostrado en la figura 15.10 usando validación cruzada dejando uno fuera con los datos de entrenamiento y luego en el conjunto de prueba. ¿Hay indicios de sobreajuste?.

15.4 Grafica las curvas ROC, y determina el AUC, obtenidas por el clasificador de la figura 15.10 para los datos de entrenamiento y para los datos de prueba.

15.5 Investiga cómo usar la función `roc.test()` del paquete `pROC` y verifica si la curva ROC generada por el clasificador de la figura 15.10 para los datos de entrenamiento es significativamente mejor que la generada para los datos de prueba.

15.6 Reconstruye el clasificador de la figura 15.10, pero ahora sin considerar los dos casos con distancia de Cook sobre 0,5. ¿Se obtiene ahora un modelo sin la presencia de datos influyentes?

15.7 Para el clasificador de la pregunta anterior, evalúa su capacidad predictiva usando validación cruzada dejando uno fuera con los datos de entrenamiento y luego en el conjunto de prueba. ¿Hay indicios de sobreajuste?.

15.8 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.

15.9 Hecho lo anterior, reconstruye el clasificador de la figura 15.10 usando validación cruzada dejando uno fuera (con todos los datos de entrenamiento) buscando la mejor sensibilidad.

15.10 De igual forma, reconstruye el clasificador usando validación cruzada de 4 pliegues buscando el mejor AUC.

15.11 Construye un clasificador para alguna variable binaria del conjunto de datos `mtcars` usando el método de todos los subconjuntos considerando modelos con dos o tres predictores. ¿Resulta confiable? ¿Qué calidad predictiva consigue? ¿Es generalizable?

15.12 Investiga de qué se trata el método Recursive Feature Elimination implementado en el paquete `caret`. Bosqueja cómo se podría buscar un clasificador utilizando este algoritmo y algún tipo de validación cruzada.

15.8 BIBLIOGRAFÍA DEL CAPÍTULO

Ayala, J. (2020). *Minería de datos*.
Consultado el 23 de junio de 2021, desde https://rpubs.com/JairoAyala/592802

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.

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/

Nelder, J. A., & Wedderburn, R. W. (1972). Generalized linear models.
Journal of the Royal Statistical Society Series A: Statistics in Society, *135*(3), 370-384.

Zelada, C. (2017). *Evaluación de modelos de clasificación*.
Consultado el 23 de junio de 2021, desde https://rpubs.com/chzelada/275494