Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Triennale in
Statistica per l'Economia e l'Impresa

Valutazione della bontà di adattamento in modelli di regressione per dati binari non raggruppati

Relatore Prof. Alessandra Salvan
Dipartimento di Scienze Statistiche

Laureanda Martha Menna
Matricola 2003670

Anno Accademico 2022/2023

Indice

Introduzione

Nell'ambito della statistica, i modelli di regressione logistica sono ampiamente utilizzati per la modellazione di variabili risposta con distribuzione binomiale. È fondamentale sviluppare metodologie appropriate per valutare i modelli stimati e per trarre conclusioni valide sulla relazione tra le variabili esplicative e la variabile risposta. Nel contesto della valutazione della bontà di adattamento di questi modelli, un aspetto critico è la presenza di dati non raggruppati, cioè dati che non sono aggregati né aggregabili. In questo caso, non sono applicabili i metodi standard per valutare la bontà di adattamento in un modello lineare generalizzato basati sulla devianza o sulla somma dei quadrati dei residui di Pearson (Agresti, 2015, paragrafo 4.4). Questa relazione si concentra su tale problema, restringendo l'attenzione al caso di osservazioni indipendenti.

In particolare questa tesi riprende concetti espressi nel capitolo 5 di Hosmer et al. (2013) e nell'articolo di Hosmer et al. (1997), considerando possibili alternative al test di Hosmer–Lemeshow, il più noto in questo ambito, come il metodo DRY-V illustrato in Lu & Yang (2018) e il test basato sui residui lisciati di le Cessie & Van Houwelingen (1991). Lo scopo di questo lavoro è fornire una panoramica completa sugli strumenti di valutazione della bontà di adattamento, per poi confrontarli mediante una simulazione.

Il capitolo 1 contiene una breve descrizione dei modelli lineari generalizzati, soffermandosi sui modelli per dati binari non raggruppati. Nel capitolo 2 si illustrano le tecniche per la valutazione della bontà di adattamento. Infine, il capitolo 3 riporta un confronto delle metodologie tramite uno studio di simulazione e successivamente l'applicazione a due insiemi di dati reali. L'elaborato si conclude con una sintesi dei risultati ottenuti.

Capitolo 1

Modelli lineari generalizzati

1.1 Introduzione

I modelli lineari generalizzati (GLM: *Generalized Linear Model*) sono un'estensione del modello di regressione lineare normale per trattare variabili risposta che non presentano distribuzione normale. Agresti (2015) fornisce una trattazione approfondita dell'argomento. L'esposizione in questo capitolo segue Salvan et al. (2020).

Siano $y_1, ..., y_n$ realizzazioni indipendenti delle variabili casuali $Y_1, ..., Y_n$ con distribuzione in una famiglia di dispersione esponenziale univariata con funzione di densità pari a

$$p(y_i; \theta_i, \phi) = \exp\left\{ \frac{\theta_i y_i - b(\theta_i)}{a_i(\phi)} + c(y_i, \phi) \right\}, \tag{1.1}$$

con $y_i \in S \subseteq \mathbb{R}$, $\theta_i \in \Theta \subseteq \mathbb{R}$, $a_i(\phi) > 0$, $i = 1, ..., n$. Il parametro θ_i è detto parametro naturale, mentre ϕ è detto parametro di dispersione.

I primi due momenti di Y_i sono rispettivamente

$$E(Y_i) = \mu_i = b'(\theta_i) = \mu(\theta_i) \tag{1.2}$$

e

$$Var(Y_i) = a_i(\phi) b''(\theta_i)\big|_{\theta_i = \theta(\mu_i)} = a_i(\phi) v(\mu_i), \tag{1.3}$$

dove $b'(\theta_i)$ e $b''(\theta_i)$ sono le prime due derivate di $b(\theta_i)$, $v(\mu_i) = b''(\theta_i)\big|_{\theta_i = \theta(\mu_i)}$ è detta funzione di varianza e $\theta(\mu_i)$ è l'inversa di $\mu(\theta_i)$. Quindi la distribuzione di Y_i è esprimibile nella seguente forma

$$Y_i \sim \ DE_1(\mu_i, a_i(\phi)v(\mu_i)), \tag{1.4}$$

con $\mu_i \in M$, lo spazio delle medie.

Le caratteristiche dei modelli lineari generalizzati sono le seguenti.

- Distribuzione della risposta: le variabili casuali indipendenti Y_i si distribuiscono come descritto nella formula (1.4)
- Predittore lineare: $\eta = X\beta$, con $\eta_i = x_i^\top \beta$ e X matrice n x p di costanti note, $p < n$ e rango pieno
- Funzione di legame: è la funzione $g(\cdot)$ che unisce μ_i a η_i, $g(\mu_i) = \eta_i$.

Quando $g(\mu_i) = \theta(\mu_i)$, si dice che $g(\cdot)$ è la funzione di legame canonica. Si indica con $f(\cdot)$ l'inversa della funzione di legame $g(\cdot)$, $f(\cdot) = g^{-1}(\cdot)$.

1.2 Inferenza basata sulla verosomiglianza nei GLM

L'inferenza nei GLM consiste nell'ottenere stime puntuali e regioni di confidenza per i parametri del modello e verificare ipotesi sul valore di questi parametri. Lo strumento che permette di fare questo è la funzione di verosomiglianza. Per maggiori dettagli si veda Salvan et al. (2020), paragrafo 2.3.

Essendo l'indipendenza una delle assunzioni dei modelli lineari generalizzati, è possibile calcolare la densità congiunta delle Y_i come il prodotto delle densità marginali e quindi la funzione di log-verosomiglianza per (β, ϕ) sarà

$$l(\beta, \phi) = \sum_{i=1}^{n} \frac{y_i\theta_i - b(\theta_i)}{a_i(\phi)} + \sum_{i=1}^{n} c(y_i, \phi) \tag{1.5}$$

con $\theta_i = \theta(\mu_i) = \theta(g^{-1}(x_i^\top \beta))$. Quando la funzione di legame è canonica, la formula (1.5) si semplifica in

$$l(\beta, \phi) = \sum_{i=1}^{n} \frac{y_i x_i^\top \beta - b(x_i^\top \beta)}{a_i(\phi)} + \sum_{i=1}^{n} c(y_i, \phi). \tag{1.6}$$

La massimizzazione richiede di calcolare la derivata prima della log-verosomiglianza, ovvero la funzione *score*, ed uguagliarla a 0, ottenendo le equazioni di verosomiglianza per β. Se ϕ è noto, le equazioni sono

$$l_r = \sum_{i=1}^{n} \frac{(y_i - \mu_i)}{Var(Y_i)} \frac{d\mu_i}{d\beta_r}, \tag{1.7}$$

con $r = 1, ..., p$.

Se $g(\cdot)$ è canonica, allora le equazioni si semplificano in

$$\sum_{i=1}^{n} \frac{1}{a_i(\phi)} y_i x_{ir} = \sum_{i=1}^{n} \frac{1}{a_i(\phi)} \mu_i x_{ir}, \tag{1.8}$$

con $r = 1, ..., p$.

Le equazioni in (1.7) possono essere riscritte in forma matriciale

$$D^\top V^{-1}(y - \mu) = 0, \tag{1.9}$$

dove $y - \mu = (y_1 - \mu_1, ..., y_n - \mu_n)^\top$, $V = \text{diag}[Var(Y_i)]$ con $i = 1, ..., n$ e D è una matrice n x p con generico elemento

$$d_{ir} = \frac{d\mu_i}{d\beta_r} = \frac{1}{g'(\mu_i)} x_{ir}, \tag{1.10}$$

con $i = 1, ..., n$, $r = 1, ..., p$. È possibile dimostrare che β e ϕ sono parametri ortogonali. Si ricava che se il legame è canonico la matrice di informazione osservata $j_{\beta\beta}$ coincide con il suo valore atteso $i_{\beta\beta}$, che in forma matriciale equivale a

$$i_{\beta\beta} = X^\top W X \tag{1.11}$$

dove

$$W = \text{diag}(w_i)$$

con $w_i = \frac{1}{(g'(\mu_i))^2 Var(Y_i)}$, $i = 1, ..., n$.

Se il legame è canonico

$$w_i = \frac{v_i(\mu_i)}{a_i(\phi)}.$$

Sfruttando la normalità asintotica dello stimatore di massima verosomiglianza

$$\hat{\beta} \dot{\sim} N_p(\beta, (X^\top W X)^{-1}), \tag{1.12}$$

sotto β per n grande.

Le equazioni di verosomiglianza (1.7) non si risolvono esplicitamente, pertanto si utilizzano metodi iterativi, come i minimi quadrati pesati iterati.

1.2.1 Devianza per ipotesi di riduzione del modello

Si partizioni il vettore β in $\beta_A = (\beta_1, ..., \beta_{p0})$ e $\beta_B = (\beta_{p0+1}, ..., \beta_p)$ e si consideri il problema di verificare $H_0 : \beta_B = 0$ contro $H_1 : \beta_B \neq 0$. Si consideri $a_i(\phi) = \phi/\omega_i$, con ϕ e i pesi ω_i noti. Sia $\hat{\beta}_0$ la stima $(\hat{\beta}_{A0}, 0)$ di β sotto H_0. In un GLM, la statistica del rapporto di verosomiglianza

$$W_P = 2(l(\hat{\beta}, \phi) - l(\hat{\beta}_0, \phi))$$

ha distribuzione asintotica nulla χ^2_{p-p0}. Si esprima la log-verosomiglianza come funzione di $\mu = (\mu_1, ..., \mu_n)^T$ e ϕ nella forma

$$l^M(\mu, \phi) = \sum_{i=1}^{n} \left\{ \omega_i \frac{y_i\theta_i - b(\theta_i)}{\phi} + c(y_i, \phi) \right\}$$

con $\theta_i = \theta(\mu_i)$. Essendo μ funzione non lineare del parametro p-dimensionale β, allora $l(\hat{\beta}, \phi) = l^M(\hat{\mu}, \phi)$. Il test del rapporto di verosimiglianza si basa sulla devianza, una misura di discrepanza tra il modello e i dati osservati. Si definisce devianza la quantità

$$D(y; \hat{\mu}) = 2\phi(l^M(y, \phi) - l^M(\hat{\mu}, \phi)). \tag{1.13}$$

Sia $l^M(\hat{\mu}, \phi)$ la log-verosomiglianza del modello saturo con $p = n$, che si ottiene ponendo $\mu_i = y_i$. La differenza $l^M(y, \phi) - l^M(\hat{\mu}, \phi)$ rappresenta una misura della diminuzione della bontà di adattamento dovuta al passaggio dal modello saturo al modello corrente con $p < n$ variabili esplicative.

1.3 Modelli per risposte binarie

In molti casi la variabile risposta si presenta in forma dicotomica, ad esempio la presenza di una malattia o meno, se una pianta è sopravvissuta o no, ecc. La soluzione è l'adattamento di un modello per dati binari.

I dati posso presentarsi in due tipi di formato. Nei dati non raggruppati le singole risposte hanno distribuzione Bernoulliana, pertanto assumono valore 1 o 0. I dati sono raggruppati quando ad ogni combinazione delle variabili esplicative corrispondono più risposte dicotomiche indipendenti e per cui si assume probabilità di successo e per ognuna viene riportato il numero totale di successi/insuccessi. È possibile dimostrare che, qualora i dati siano esprimibili sia in formato raggruppato che non raggruppato, le funzioni di verosomiglianza per β nei due casi sono equivalenti, quindi le stime e

gli *standard error* coincidono. In questa tesi si tratterà solo di dati in formato non raggruppato, e per cui non sia possibile passare alla forma raggruppata, ad esempio per la presenza di esplicative quantitative continue con valori non ripetuti.

1.3.1 Modello logistico

Siano $y_1, ..., y_n$ realizzazioni di variabili indipendenti Y_i, con distribuzione binomiale, $Bi(1, \pi_i)$. Riprendendo la notazione usata nel paragrafo 1.1,

$$Y_i \sim \ DE_1(\pi_i, \pi_i(1-\pi_i)),$$

dove π_i è la media della i-esima variabile.

La funzione di legame più utilizzata risulta essere la *logit*, da qui l'espressione modello logistico. Essa rappresenta il legame canonico e si esprime come

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = x_i^\top \beta, \tag{1.14}$$

da cui è possibile ricavare

$$\pi_i = \frac{\exp(\sum_{r=1}^{p} \beta_r x_{ir})}{1 + \exp(\sum_{r=1}^{p} \beta_r x_{ir})}. \tag{1.15}$$

Uno dei vantaggi dell'utilizzo del modello logistico è senza dubbio l'interpretazione dei parametri utilizzando il logaritmo della quota (*log-odds*). La quota $\pi_i/(1-\pi_i)$ è il rapporto tra probabilità di successo e insuccesso. La variazione unitaria di un generico x_{ir} a parità delle rimanenti variabili (e in assenza di interazioni) comporta la variazione moltiplicativa della quota pari a $\exp(\beta_r)$.

In un modello con una sola variabile esplicativa dicotomica le osservazioni sulla risposta binaria sono suddivise in due gruppi, ad esempio di soggetti fumatori e non, tale che $x_i = 1$ se l'i-esimo soggetto appartenente al gruppo dei fumatori e $x_i = 0$ se non è fumatore. In questo caso, il modello logistico si basa sull'assunzione

$$Pr(Y_i = 1) = Pr(Y_i = 1|x_i) = \frac{\exp(\beta_1 + \beta_2 x_i)}{1 + \exp(\beta_1 + \beta_2 x_i)}, \tag{1.16}$$

dove

$$Pr(Y_i = 1|x_i = 1) = \frac{\exp(\beta_1 + \beta_2)}{1 + \exp(\beta_1 + \beta_2)}$$

e

$$Pr(Y_i = 1|x_i = 0) = \frac{\exp(\beta_1)}{1 + \exp(\beta_1)}.$$

Il log-rapporto delle quote è

$$\log\left(\frac{Pr(Y_i=1|x_i=1)/Pr(Y_i=0|x_i=1)}{Pr(Y_i=1|x_i=0)/Pr(Y_i=0|x_i=0)}\right)=\beta_2.$$

La quota per i soggetti fumatori è $\exp(\beta_2)$ volte la quota per i non fumatori.

1.3.2 Devianza

La devianza nei modelli binari con dati non raggruppati corrisponde a

$$D(y;\hat{\pi})=-2\sum_{i=1}^{n}[y_i\log(\hat{\pi}_i)+(1-y_i)\log(1-\hat{\pi}_i)],$$

con y_i che può valere solo 0 oppure 1.

Mentre nei modelli con dati raggruppati la devianza può essere utilizzata come strumento per valutare la bontà di adattamento del modello, ciò non è possibile nei modelli con dati non raggruppati. In questo caso la devianza viene utilizzata solo per il confronto di due modelli annidati, sfruttando la distribuzione approssimata $\chi^2_{p-p_0}$, dove p rappresenta il numero delle variabili esplicative del modello corrente, mentre p_0 del modello ridotto. Maggiori dettagli sul modello logistico sono riportati nel capitolo 5 del volume Agresti (2013).

Capitolo 2

Valutazione della bontà di adattamento

2.1 Introduzione

Un modello può essere valutato per due aspetti: capacità di discriminazione e bontà di adattamento. La discriminazione è la capacità di distinguere correttamente tra le diverse categorie o classi di output. Questa informazione si ricava costruendo la curva ROC. La bontà di adattamento quantifica l'accuratezza delle probabilità stimate della variabile risposta e tra i test più usati per ottenere questa informazione vi è il test di Hosmer–Lemeshow (HL), introdotto in Hosmer & Lemeshow (1980). Questo capitolo tratta gli strumenti utilizzati per valutare la bontà di adattamento nei modelli di regressione logistica con dati non raggruppati, riprendendo concetti illustrati nel capitolo 5 di Hosmer et al. (2013).

2.2 Curva ROC

La curva ROC rappresenta uno strumento grafico per valutare la capacità predittiva di un modello logistico. Si costruisce una tabella di contingenza di dimensioni 2 x 2, detta tabella di classificazione, incrociando i risultati della risposta osservati, y_i, con quelli di una variabile dicotomica, $\hat{y}_i$, i cui valori derivano dalle probabilità stimate dal modello, $\hat{\pi}_i$ per $i = 1, ..., n$. Stabilita una soglia π_0, solitamente pari a 0.5, la variabile dicotomica è tale che

$$\hat{y}_i = \begin{cases} 1 & \text{se } \hat{\pi}_i \geq \pi_0 \\ 0 & \text{altrimenti.} \end{cases}$$

Si riassume nella Tabella 2.1.

TABELLA 2.1: Tabella di classificazione

	$\hat{y}=0$	$\hat{y}=1$	
$y=0$	n_{00}	n_{01}	n_{0+}
$y=1$	n_{10}	n_{11}	n_{1+}
	n_{+0}	n_{+1}	n

Dalla Tabella 2.1 si ricavano le stime di due quantità, la sensibilità e la specificità. La sensibilità pari a

$$Pr(\hat{Y}_i = 1|Y_i = 1),$$

stimabile con n_{11}/n_{1+}, rappresenta il tasso di veri positivi (TRP, *True Positive Rate*). La specificità equivale a

$$Pr(\hat{Y}_i = 0|Y_i = 0),$$

stimabile con n_{00}/n_{0+}, rappresenta il complemento a 1 del tasso di falsi positivi (FPR, *False Positive Rate*).

La curva ROC (*Receiver Operating Characteristic*) è il grafico dei valori stimati delle coppie (FPR,TPR) al variare della soglia π_0, che assume valori fra 1 a 0. Se π_0 tende a 1 sia TPR che FPR sono prossimi a 0. Invece, se π_0 tende a 0, le due quantità sono prossime a 1. Fissata la specificità, la capacità predittiva del modello è tanto maggiore quanto più grande è la sensibilità. L'area sotto la curva prende il nome di AUC (*Area Under the Curve*) e assume valori compresi tra 0 e 1. Se $AUC = 0.5$, allora il modello stimato avrà una pessima capacità predittiva e la curva ROC è data dal segmento che unisce l'origine e il punto (1,1), al contrario se $AUC = 1$ il modello stimato è ottimo e $\hat{\pi}_i = y_i$ per ogni osservazione.

2.3 Test di Hosmer–Lemeshow

Il test di Hosmer-Lemeshow rappresenta uno strumento per la valutazione della bontà di adattamento di un modello basato su un test chi-quadrato. Hosmer & Lemeshow (1980) hanno proposto un metodo che verifica se le probabilità stimate con il modello sono coerenti con i valori osservati della risposta. Il test aiuta a determinare se vi è evidenza di mancanza di adattamento o scostamento dal modello di regressione logistica ipotizzato.

La statistica test viene costruita suddividendo i dati in due modi possibili. In particolare si possono utilizzare:

- quantili delle probabilità stimate,
- valori fissati delle probabilità.

I quantili sono indici di posizione che dividono una distribuzione ordinata in s parti uguali. Con il primo metodo, le osservazioni vengono suddivise in G gruppi sulla base dei quantili della distribuzione delle probabilità stimate, quindi il primo gruppo avrà le osservazioni con le probabilità stimate più piccole e l'ultimo le osservazioni con le probabilità stimate più grandi. Il secondo metodo utilizza $G = 10$ gruppi definiti sulla base di valori di soglia fissati di livello $k/10$, $k = 1, 2, ..., 9$, e i gruppi corrispondono alle probabilità stimate tra i valori di soglia adiacenti. Ad esempio, il primo gruppo contiene tutte le osservazioni con probabilità stimata minore o uguale a 0.1, mentre il decimo gruppo contiene le osservazioni che hanno probabilità stimata maggiore di 0.9. La statistica test viene calcolata per entrambe le strategie nel medesimo modo. Nei pacchetti dei *software*, come nella libreria R DescTools (Signorell et al., 2023), per differenziare le due strategie di raggruppamento la statistica basata sul primo modo viene indicata con $\hat{C}$ e quella basata sul secondo $\hat{H}$. In questo paragrafo per semplicità si userà il simbolo $\hat{C}$ per riferirsi alla statistica test generale.

Sia y_{ij} la risposta binaria dell'osservazione j nel gruppo i, $i = 1, ..., G$, $j = 1, ..., n_i$. Sia $\hat{\pi}_{ij}$ la corrispondente probabilità stimata per il modello stimato relativo ai dati non raggruppati. Si consideri il problema di verificare $H_0 : \pi_{ij} = \pi^0_{ij}$ contro $H_1 : \pi_{ij} \neq \pi^0_{ij}$ per $i = 1, ..., G$, $j = 1, ..., n_i$, dove $\pi^0_{ij} = g^{-1}(x_{ij}^\top \beta)$ e $g(\cdot)$ è la funzione *logit*. Per calcolare la statistica test $\hat{C}$, le osservazioni vengono suddivise in G gruppi di pari numerosità, solitamente $G = 10$. La statistica è

$$\hat{C} = \sum_{i=1}^{G} \frac{(\sum_{j=1}^{n} y_{ij} - \sum_{j=1}^{n} \hat{\pi}_{ij})^2}{(\sum_{j=1}^{n} \hat{\pi}_{ij})[1 - (\sum_{j=1}^{n} \hat{\pi}_{ij})/n_i]}. \tag{2.1}$$

Sotto H_0, Hosmer & Lemeshow (1980) mostrano che la statistica $\hat{C}$ è approssimabile ad una distribuzione chi-quadrato χ^2_{G-2}, dove $G - 2$ sono gradi di libertà, al contrario sotto H_1 la distribuzione approssimata è un chi-quadrato non centrale $\chi^2_{G-2,\lambda}$, dove i gradi di libertà sono gli stessi e λ rappresenta un parametro di non centralità (si veda l'Appendice A per maggiori dettagli su questa distribuzione).

Studi aggiuntivi illustrati in Hosmer et al. (1988) hanno dimostrato che la prima strategia di raggruppamento risulta preferibile alla seconda, in quanto vi è una maggiore aderenza alla distribuzione χ^2_{G-2}, specialmente quando molte delle probabilità stimate

sono prossime a zero. Spesso ci si riferisce ai gruppi con l'espressione "decili del rischio", nata nell'ambito degli studi medici, dove la risposta $y = 1$ indica la presenza di una malattia.

Nel cap. 5 di Hosmer et al. (2013) si sottolinea come il vantaggio di questo test sia la facilità di calcolo e di interpretazione, mentre lo svantaggio è l'influenza sul risultato del modo in cui i dati sono suddivisi. Questo può causare interpretazioni fuorvianti, difatti è possibile classificare nello stesso gruppo osservazioni con valori delle variabili esplicative molto diversi.

2.3.1 Modifica del test HL

Il test HL tradizionale riesce a cogliere anche discrepanze irrilevanti tra le probabilità stimate e le vere probabilità all'aumentare della numerosità campionaria, in quanto la potenza del test cresce al crescere della numerosità campionaria e questo può causare il rifiuto di modelli non perfetti, ma accettabili. Nattino et al. (2020) hanno proposto un metodo per risolvere il problema, basato sulla costruzione di un parametro standardizzato di non centralità, che misura la bontà di adattamento, ed è indipendente da n.

Il parametro di non centralità λ risulta direttamente proporzionale alla numerosità, di conseguenza si introduce la sua versione standardizzata

$$\epsilon = \sqrt{\frac{\lambda}{n}}. \tag{2.2}$$

La formula (2.2) fornisce un parametro per fare inferenza sulla bontà di adattamento del modello. Il parametro λ non è noto e viene stimato con $\hat{\lambda} = \max\left\{\hat{C} - (G-2), 0\right\}$, da qui si ottiene lo stimatore

$$\hat{\epsilon} = \sqrt{\frac{\max\left\{\hat{C} - (G-2), 0\right\}}{n}}. \tag{2.3}$$

Se il modello si adatta perfettamente ai dati, la statistica $\hat{C}$ ha distribuzione chi-quadrato centrale, quindi $\lambda = 0$ e $\epsilon = 0$. Ad un peggioramento del modello corrisponde una crescita di ϵ. Grazie alla standardizzazione rispetto alla dimensione del campione, ϵ dipende solo da quanto il modello si scosta dal perfetto adattamento.

Si consideri il problema di verificare $H_0 : \epsilon = 0$ contro $H_1 : \epsilon > 0$. La statistica test rifiuta H_0 con livello di significatività pari a α se lo 0 non appartiene all'intervallo

di confidenza unilaterale per ϵ, la cui costruzione viene illustrata nel paragrafo 3.1 di Nattino et al. (2020). Questo test è equivalente al test HL tradizionale.

Si può tuttavia modificare il problema fissando una soglia per ϵ, pari a ϵ_0, entro la quale il modello risulta accettabile e si verifica l'ipotesi $H_0 : \epsilon \leq \epsilon_0$. Il livello di significatività osservato del test, o *p-value*, è calcolabile comparando il valore della statistica HL tradizionale $\hat{C}$ alla distribuzione chi-quadrato non centrale con parametro di non centralità $\epsilon_0^2 n$ e $G-2$ gradi di libertà. Il valore di ϵ_0 può essere fissato a

$$\epsilon_0 = \sqrt{\frac{\chi^2_{\lambda=0,df=G-2,\alpha=0.05} - (G-2)}{n_0}}, \tag{2.4}$$

dove $n_0 = 10^6$. Nella Figura 2.1 si mettono a confronto i *p-value* dei test al variare della numerosità campionaria. I due test forniscono risultati simili per campioni piccoli o medi, mentre la differenza aumenta quando il campione è molto numeroso. In particolare il *p-value* del test proposto è maggiore di quello del test tradizionale, come desiderato per far sì che modelli non perfetti, ma accettabili, non siano rifiutati. La metodologia proposta consente di sostituire interamente il test di Hosmer-Lemeshow, per qualsiasi dimensione campionaria, dal momento che si basano sulle stesse assunzioni. La Figura 2.1 è stata realizzata con il *software* R e il codice è riportato nell'Appendice C.

2.4 Test di le Cessie-van Houwelingen

Sfruttando metodi non parametrici basati sulla stima *kernel*, le Cessie & Van Houwelingen (1991) hanno proposto un test basato sulla somma dei quadrati dei residui lisciati, il cui *p-value* viene calcolato usando un'approssimazione della normale o un chi-quadrato scalato. Il lisciamento *kernel* è una tecnica statistica per stimare un valore reale come media pesata dei valori vicini osservati. Il peso è definito dal *kernel*, tale che i punti più vicini abbiano peso maggiore. La funzione stimata è lisciata e il livello di lisciamento è dato dal parametro h, detto *bandwidth* (ampiezza di banda). Esiste una vasta gamma di contributi per questo argomento, tra cui ad esempio Gasser & Müller (1979).

L'obiettivo del test è in particolare valutare se la funzione $f(\cdot)$, l'inversa della funzione di legame, coincide con una specifica funzione $f_0(\cdot)$. Nel caso del modello logistico con una sola variabile esplicativa, $f_0(\cdot)$ corrisponde alla formula (1.16). Siano

$$r_s(x_i) = \frac{y_i - f_0(x_i)}{\sqrt{f_0(x_i)(1-f_0(x_i))}}$$

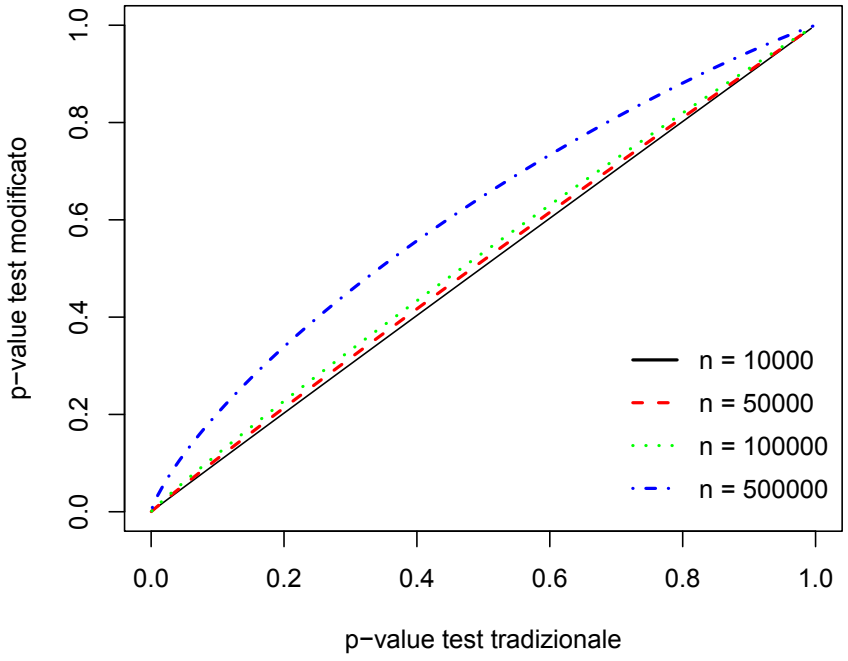

FIGURA 2.1: Relazione tra il *p-value* del test HL tradizionale e il *p-value* del test proposto per diverse numerositá campionari.

i residui del modello, $y_i - f_0(x_i)$, standardizzati. Per costruire la statistica test occorre calcolare i residui standardizzati lisciati

$$\tilde{r}(x_i) = \frac{\sum_{j=1}^{n} r_s(x_j) K[(x_i - x_j)/h_n]}{\sum_{j=1}^{n} K[(x_i - x_j)/h_n]},$$

con $i, j = 1, ..., n$, che si ottengono mediante la stima *kernel* di Nadaraya (1964) e Watson (1964), Il parametro *bandwidth* h_n dipende dalla dimensione campionaria. La funzione K è una funzione *kernel* non negativa simmetrica, con vincoli $\int K(z)dz = 1$ e $\int K(z)^2 dz = 1$. I residui standardizzati lisciati $\tilde{r}(x_i)$ sono una media pesata dei residui che si trovano intorno ad x_i, dove la funzione K determina i pesi e il parametro *bandwidth* determina l'ampiezza dell'intervallo in cui prendere i residui pesati.

Si ricava che la media di $\tilde{r}(x_i)$ per ogni $i = 1, ..., n$ è

$$E(\tilde{r}(x_i)) = 0$$

e, assumendo l'indipendenza delle osservazioni, la varianza è

$$var(\tilde{r}(x_i)) = \frac{\sum_{j=1}^{n} K[(x_i - x_j)/h_n]^2}{\{\sum_{j=1}^{n} K[(x_i - x_j)/h_n]\}^2}.$$

La statistica test T è definita come

$$T = \frac{1}{n}\sum_{i=1}^{n} \tilde{r}(x_i)^2 v(x_i), \tag{2.5}$$

dove

$$v(x_i) = \frac{\{\sum_{j=1}^{n} K[(x_i - x_j)/h_n]\}^2}{\sum_{j=1}^{n} K[(x_i - x_j)/h_n]^2}$$

è il reciproco della varianza del residuo standardizzato lisciato relativo a x_i. Solitamente la scelta della funzione *kernel* non risulta essere così importante, infatti non vi sono evidenze di alterazioni dei risultati al variare di questa. La più semplice da trattare è la densità della distribuzione uniforme su $[-\frac{1}{2}, \frac{1}{2}]$. Al contrario, la scelta del *bandwidth* è cruciale, poiché se viene fissato troppo piccolo la statistica T ha scarsa potenza e se troppo grande tutte le deviazioni locali vengono eliminate. Si suggerisce di sceglierlo in modo tale che ogni regione dove i residui sono mediati contenga approssimativamente $\sqrt{n}$ osservazioni.

Nella formula (2.5) la statistica viene costruita assumendo che la funzione $f_0(\cdot)$ sia nota. Si consideri ora il caso in cui $f_0(\cdot)$ sia stimata. Si può scrivere $f_0(\cdot)$ come

$$f_0(x) = f(x, \beta_0),$$

dove f è una funzione conosciuta e β_0 è un vettore colonna di parametri sconosciuti che vengono stimati con il metodo di massima verosomiglianza, $\hat{\beta}$. Quindi $\hat{f(x)} = f(x, \hat{\beta})$. La statistica è

$$\hat{T} = \frac{1}{n}\sum_{i=1}^{n} \tilde{r_e}(x_i)^2 v(x_i), \tag{2.6}$$

dove i residui standardizzati lisciati $\tilde{r_e}$ sono costruiti usando il valore stimato $\hat{g}(x_i)$. Sotto condizioni di regolarità, $\hat{T}$ è asintoticamente equivalente a T. Al tempo stesso, per campioni finiti la media e la varianza di $\hat{T}$ sono più piccole della media e della varianza di T, come illustrato nel paragrafo 6 di le Cessie & Van Houwelingen (1991).

La distribuzione di $\hat{T}$ non è nota, quindi i suoi quantili sono comparati con quelli di una distribuzione normale e di una distribuzione chi-quadrato $c\chi^2_v$ per scegliere l'approssimazione migliore. La chi-quadrato scalata con v gradi di libertà è usata per le approssimazioni di distribuzioni di variabili casuali non negative. Le costanti c e v sono

ottenute attraverso le equazioni della media e varianza di $\hat{T}$, $c = Var(\hat{T})/(2E(\hat{T}))$ e $v = 2E(\hat{T})^2/Var(\hat{T})$. Alcune simulazioni in le Cessie & Van Houwelingen (1991) hanno dimostrato che, per variabili continue, la distribuzione chi-quadrato scalata fornisce approssimazioni migliori, soprattutto per bassi livelli di significatività.

È possibile estendere questo approccio al caso con più variabili esplicative, dove con p dimensioni si utilizza una funzione *kernel* moltiplicativa $K(z)$, definita per $z = (z_1, ..., z_p)$ come

$$K(z) = \prod_{l=1}^{p} K(z_l),$$

dove K è la funzione unidimensionale definita sopra e h_l è il *bandwidth* della l-esima covariata. Si definisce $h_l = h_n/s_l$, dove s_l è la deviazione standard della l-esima variabile esplicativa. In questo caso il numero di parametri *bandwidth* si riduce a 1 e h_n è il *bandwidth* standardizzato.

Quando sono presenti sia variabili esplicative continue che discrete, uno degli approcci suggeriti nel paragrafo 8 di le Cessie & Van Houwelingen (1991) consiste nel considerare solo le variabili continue ed ignorare l'effetto delle categoriali. L'approccio è corretto se i problemi dell'adattamento del modello derivano solo dalle variabili continue e se non vi sono termini di interazione tra le variabili del modello. L'articolo riporta altri approcci possibili non illustrati qui.

Studi successivi degli stessi autori hanno portato alla formulazione di un altro test per la bontà di adattamento applicabile a modelli logistici con effetti casuali. Questi modelli assumono l'esistenza di caratteristiche non osservabili comuni a tutte le osservazioni inerenti la stessa unità. La statistica test ottenuta risulta simile a quella illustrata. Per una trattazione più estesa si veda le Cessie & Van Houwelingen (1995).

2.5 Stima non parametrica basata sulla *cross-validation*

La valutazione di un modello di regressione per risposte binarie è un tema cruciale, studiato con diversi approcci. Uno di questi, introdotto da Lu & Yang (2018), si basa su una stima non parametrica basata su un metodo di validazione incrociata, in inglese *cross-validation* (CV). La CV è uno strumento di *machine learning* utilizzato per stimare l'accuratezza predittiva di un modello, per identificare il miglior candidato tra i metodi o per selezionare un parametro di penalizzazione di una procedura statistica. La consistenza della validazione incrociata per confrontare modelli è stata esaminata da Zhang & Yang (2015). In questo paragrafo si illustra l'impiego della CV con votazione

per valutare la bontà di adattamento di un modello per risposte binarie. Il modello stimato viene confrontato con una stima non parametrica della funzione

$$f(x_i) = P(Y_i = 1|X_i),$$

con $i = 1, ..., n$. I metodi non parametrici considerati sono il *random forest* e il *bagging*. Il *random forest* è un metodo di apprendimento automatico sia per problemi di classificazione che di regressione. È molto popolare grazie alla sua elevata accuratezza e alla capacità di gestire grandi insiemi di dati con un numero elevato di predittori, quindi ampiamente utilizzato in vari campi applicativi, tra cui la bioinformatica, l'elaborazione del linguaggio naturale. Si tratta di un'estensione del *bagging* (*Bootstrap Aggregating*). Un riferimento per questi metodi è Hastie et al. (2009).

La *cross-validation* prevede di lavorare su due porzioni dei dati, l'insieme di stima e l'insieme di verifica. L'insieme di stima viene usato per adattare il modello, mentre sull'insieme di verifica si valutano le prestazioni del modello stimato. Più in dettaglio, l'insieme di dati originario dovrebbe essere suddiviso in k parti con uguale numerosità, in modo tale che iterativamente venga utilizzata una porzione come insieme di stima e le altre $k - 1$ come insieme di verifica. Il risultato sarà una stima più accurata delle prestazioni, in quanto ogni volta la parte di dati su cui si verifica l'adattamento del modello è diversa.

2.5.1 Metodo DRY-V

I dati sono suddivisi in due blocchi con numerosità n_1 e $n_2 = n - n_1$. La porzione di insieme di stima è $z_1 = (x_i, y_i)_{i=1}^{n_1}$ e quella di insieme di verifica $z_2 = (x_i, y_i)_{i=n_1+1}^{n}$. Si adatta il modello parametrico su z_1 ottenendo la stima $\hat{f}_{n_1,1}(x)$, analogamente con il metodo non parametrico, in questa tesi si utilizzano il *random forest* o il *bagging*, si calcola la stima $\hat{f}_{n_1,2}(x)$. Quindi si calcola la log-verosomiglianza predittiva su z_2

$$CV(\hat{f}_{n_1,j}) = \sum_{i=n_1+1}^{n} y_i \log \hat{f}_{n_1,j}(x_i) + (1 - y_i) \log(\hat{f}_{n_1,j}(x_i)), \tag{2.7}$$

$j = 1, 2$.

Se $CV(\hat{f}_{n_1,1}) < CV(\hat{f}_{n_1,2})$, allora la stima non parametrica è preferibile a quella parametrica. Sia δ una permutazione delle osservazioni e $CV_\delta(\hat{f}_{n_1,j})$ il valore del criterio definito in (2.7) calcolato dopo aver effettuato la permutazione.
Sia m il numero di permutazioni casuali, se $CV_{\delta i}(\hat{f}_{n_1,1}) \geq CV_{\delta i}(\hat{f}_{n_1,2})$, allora $\tau_{\delta i} = 1$, altrimenti $\tau_{\delta i} = 0$, con $i = 1, ..., m$. Se $\sum_{i=1}^{m} \tau_{\delta i} \geq \frac{m}{2}$, allora si accetta il modello

parametrico.

Questo metodo prende il nome di DRY-V (dall'inglese ***d**ivide, **r**e-fit, say **y**es or no with **v**oting*). Questo criterio si differenzia dalla *cross-validation* tradizionale, dove sarebbe stata fatta una media dei valori provenienti da diverse suddivisioni dei dati prima di comparare i diversi metodi.

Si dimostra teoricamente che le probabilità di errore di primo e secondo tipo convergono a 0 all'aumentare della dimensione del campione. Pertanto il metodo DRY-V conduce asintoticamente al risultato corretto. Inoltre, uno dei vantaggi del metodo DRY-V rispetto ai test HL e di le Cessie–van Houwelingen è la capacità di individuare maggiormente problemi dovuti a variabili mancanti, non essendoci il prerequisito della presenza di tutte le variabili esplicative. Al tempo stesso, il metodo presenta delle debolezze. La prima consiste nella possibilità di applicazione solo per osservazioni indipendenti e identicamente distribuite. Sarebbero necessari ulteriori studi per poter trattare anche dati longitudinali. Questo limite è presente anche negli altri test esposti. Inoltre, per qualsiasi metodo basato sulla CV si opera su un campione ridotto, che potrebbe portare a conclusioni diverse rispetto a quelle a cui si arriverebbe usando l'intero campione.

Al contrario dei metodi citati nei paragrafi precedenti, questo approccio basato su strumenti di *machine learning* può trattare dati ad alta dimensionalità. Per maggiori approfondimenti in questa direzione si veda Janková et al. (2020).

2.6 Altri possibili test

Questo paragrafo riporta ulteriori test, che insieme al test HL e al test di le Cessie-van Houwelingen sono confrontati nell'articolo Hosmer et al. (1997).

Test di Stukel

Un modello per valutare l'adeguatezza del modello logistico è stato proposto da Stukel (1988). Il test valuta l'adeguatezza del modello verificando che il predittore lineare abbia una forma coerente con la funzione logistica. Stukel introduce una classe di modelli lineari generalizzati con funzione di legame

$$\text{logit}(\pi) = x^T\beta + \alpha_1 z_1 + \alpha_2 z_2$$

con due parametri di regressione aggiuntivi, α_1 e α_2, e le variabili $z_1 = \frac{1}{2}(x^T\beta)^2 I(x^T\beta \geq 0)$ e $z_2 = -\frac{1}{2}(x^T\beta)^2 I(x^T\beta < 0)$. $I(\cdot)$ è la funzione indicatrice, che assume valore 1 se

l'argomento è vero, 0 altrimenti.

Nel modello lineare logistico usuale $\alpha_1 = 0$ e $\alpha_2 = 0$. L'Autore propone come test di bontà di adattamento un test *score* per valutare l'ipotesi di nullità dei parametri α_1 e α_2 fornendo una statistica test, indicata con $\hat{ST}$, la cui distribuzione sotto H_0 è approssimabile ad un chi-quadrato con 2 gradi di libertà.

Test di Royston

Si considerano due procedure sviluppate in Royston (1992) e in Royston et al. (1993), che sono in grado di individuare scostamenti nella funzione di legame. Il primo test ha lo scopo di individuare nella funzione di legame lo scostamento dalla monotonicità. La relativa statistica è

$$\hat{PR}_1 = \max_{1 \leq l \leq n} |q_l|,$$

dove $q_l = -\sum_{i=1}^{l}(y_{(i)} - \hat{\pi}_{(i)})$, $\hat{\pi}_{(i)}$ rappresenta la i-esima probabilità stimata ordinata in senso crescente e $y_{(i)}$ è il valore della variabile risposta associato ad essa. Il test prende il nome di "Royston monotone". Il secondo test è

$$\hat{PR}_2 = \max_{1 \leq l \leq n/2} |q_l - q_{n-l}|$$

e viene indicato con "Royston quadratic", in quanto individua la possibile presenza di un termine quadratico nella funzione di legame. Royston effettua una trasformazione sulle due quantità in modo da poterle approssimare ad una distribuzione normale standard, come illustrato nell'Appendice di Royston (1992).

Capitolo 3

Simulazione e applicazione a dati reali

3.1 Introduzione

In questo capitolo si confrontano attraverso una simulazione i comportamenti dei test di bontà di adattamento introdotti nel capitolo 2, esaminando inizialmente l'adeguatezza della distribuzione teorica nulla della statistica test ed in secondo luogo la capacità dei test di individuare possibili scostamenti dal modello logistico, come la presenza di un termine quadratico. Si valutano separatamente le prestazioni del metodo DRY-V. Lo studio è stato svolto utilizzando il *software* R e i seguenti pacchetti: "DescTools" sviluppato in Signorell et al. (2023) per calcolare le statistiche del test HL, "LogisticDx" sviluppato in Dardis (2021) per calcolare il test di Stukel, "pROC" sviluppato in Robin et al. (2023) per costruire la curva ROC, "tidyverse" e "MASS" nel metodo DRY-V illustrati rispettivamente in Wickham et al. (2023) Venables & Ripley (2023) e "randomForest" in Breiman & Cutler (2023) per calcolare le stime non parametriche con i metodi *random forest* e *bagging*. Le parti di codice relative ai test di Royston e di le Cessie–van Houwelingen sono state implementate in questa relazione. Infine, si riportano le applicazioni a due insiemi di dati reali, tratti il primo dal capitolo 1 del volume Hosmer et al. (2013) e il secondo dal capitolo 4 di Agresti (2013). Il codice R è riportato nell'Appendice C.

3.2 Risultati di simulazione

Nella prima parte della simulazione le statistiche considerate sono: la statistica di Hosmer–Lemeshow, calcolata sia usando i quantili delle probabilità stimate, $\hat{C}$, che i valori fissati, $\hat{H}$, la statistica test di Stukel, $\hat{ST}$, le due versioni del test di Royston, $\hat{PR}_1$ e $\hat{PR}_2$, ed infine il test di le Cessie–van Houwelingen, $\hat{T}$. I valori dei coefficienti delle variabili usati per generare i dati sono gli stessi utilizzati in Hosmer et al. (1997). Il primo obiettivo della simulazione è verificare la probabilità di errore di primo tipo. Il modello utilizzato per la generazione dei dati è

$$\text{logit}(\pi_i) = \beta_0 + \beta_1 x_i,$$

dove $\beta_0 = 1.40$, $\beta_1 = 0.85$ e $i = 1, ..., n$. La varibile esplicativa x_i, i cui valori sono generati una sola volta per ciascun ciclo di simulazioni, si distribuisce come un'uniforme nell'intervallo [-6,6]. I campioni generati hanno numerosità pari a n, dove $n = 100, 200, 300, 400, 500$, e, per ogni scenario, si realizzano 1000 replicazioni con generazione Monte Carlo. La variabile risposta è

$$y_i = \begin{cases} 1 & \text{se } u_i \leq \pi_i \\ 0 & \text{altrimenti.} \end{cases},$$

dove u_i si distribuisce come un'uniforme nell'intervallo [0,1], π_i è definito nella formula (1.15), $i = 1, ..., n$. Il *p-value* della statistica di le Cessie-van Houweligen è calcolato utilizzando le stime della media e della varianza ottenute tramite simulazione, invece che le approssimazioni delle due quantità utilizzate nell'articolo Hosmer et al. (1997). Per il test HL è possibile che in alcune situazioni il numero dei gruppi sia inferiore a 10.

La Tabella 3.1 mostra la percentuale di volte che i test rifiutano l'ipotesi nulla al livello nominale $\alpha = 0.05$. I test di Royston non rifiutano mai l'ipotesi nulla e questo indica la necessità di ulteriori studi per il calcolo dei *p-value*, come suggerito in Hosmer et al. (1997). Le altre statistiche rifiutano con livello effettivo più prossimo a quello nominale del 5%.

Si valuta ora la potenza dei test nell'individuare la presenza di un termine quadratico. I dati sono generati mediante il modello logistico

$$\text{logit}(\pi_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2,$$

dove $\beta_0 = 1.40$, $\beta_1 = 0.85$, $\beta_2 = -0.11$ e $i = 1, ..., n$. La Figura 3.1 mostra che tutti i

TABELLA 3.1: Percentuale di rifiuto simulata usando campioni di numerosità 100, 200, 300, 400 e 500 con 1000 replicazioni al livello $\alpha = 0.05$

statistica/numerosità	100	200	300	400	500
$\hat{C}$	4.4	8.2	6.2	6.8	6.3
$\hat{H}$	4.8	5.1	5.5	5.4	4.3
$\hat{ST}$	4.6	5.4	4.5	5.4	5.5
$\hat{PR}_1$	0	0	0	0	0
$\hat{PR}_2$	0	0.1	0	0	0
$\hat{T}$	2.9	3.1	3	4.6	4.3

test hanno prestazioni migliori al crescere della numerosità campionaria e tutti, eccetto uno, individuano la maggior parte delle volte la non adeguatezza del modello stimato. Invece, il test HL calcolato raggruppando i dati in base a valori fissati delle probabilità mostra i risultati peggiori, con basse percentuali di rifiuto dell'ipotesi nulla. I test sono stati valutati a parità di α effettivo.

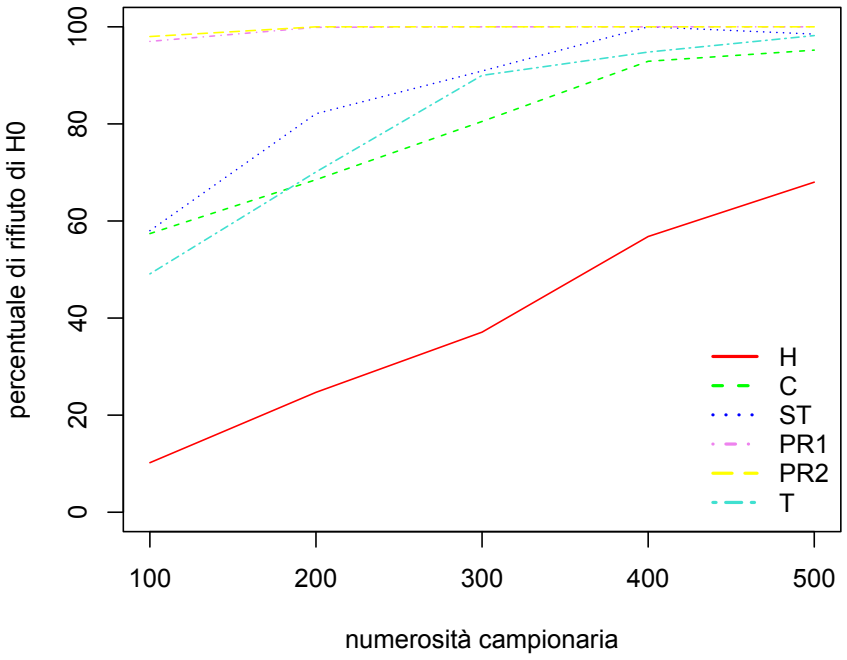

FIGURA 3.1: Percentuale di rifiuto simulata usando campioni di numerosità 100, 200, 300, 400 e 500 con 1000 replicazioni al livello effettivo $\alpha = 0.05$

Sulla base dei risultati ottenuti si evince che il test preferibile è il test di Stukel, il

quale mostra una buona capacità sia quando il modello stimato è corretto sia quando deve individuare la presenza di un termine quadratico nel predittore lineare del modello stimato. Va tuttavia sottolineato che l'ipotesi alternativa considerata, l'aggiunta di un termine quadratico, è quella per cui è stato costruito il test di Stukel.

Si considera ora la valutazione del metodo DRY-V basato su uno dei due metodi non parametrici, *random forest* o *bagging*. Anche per questo metodo si valuta prima la probabilità di errore di primo tipo e successivamente l'abilità nell'individuare la presenza di uno o più termini nel predittore lineare, considerando sia lo scenario in cui viene meno solo un termine sia lo scenario in cui nel modello logistico stimato mancano diverse covariate. In tutte le analisi svolte per questo metodo si riprendono i dati usati in Lu & Yang (2018).

I dati sono generati dal modello logistico con predittore lineare

$$\eta_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i} \beta_5 x_{5i} + \beta_6 x_{6i},$$

dove $\beta_0 = 0.5$, $\beta_1 = 4$, $\beta_2 = 2$, $\beta_3 = 1$, $\beta_4 = -0.5$, $\beta_5 = -1$, $\beta_6 = 3$ e $i = 1, ..., n$. Le prime quattro variabili esplicative si distribuiscono come normali standard indipendenti, mentre le ultime due sono indipendenti dalle prime quattro e hanno distribuzione normale bivariata con $Cov(X_5, X_6) = 0.5$. Le esplicative sono generate una sola volta per ogni ciclo di simulazione. Per i metodi *random forest*, indicato con "RF", e *bagging*, indicato con "BAG", sono state considerate tutte e sei le variabili esplicative in ogni scenario. Tuttavia, nel caso in cui si eliminassero possibili covariate non significative, i due metodi non parametrici avrebbero addirittura comportamento migliore. In tutte le simulazioni si genera un campione di numerosità n, con $n = 100, 200, 300$ e per ognuno di questi si realizzano 100 replicazioni. Il numero di replicazioni è scelto in modo tale che i tempi computazionali non siano eccessivi. Questo metodo, come illustrato nel paragrafo 2.5.1, si basa sul confronto della stima parametrica con la stima non parametrica.

Per esaminare la probabilità di errore di primo tipo, si adatta un modello di regressione logistica contenente tutte e sei le variabili esplicative. Dalla Tabella 3.2 si evince la buona abilità del metodo DRY-V nel selezionare il modello parametrico cha ha effettivamente generato i dati sia usando il *random forest* che il *bagging*. Tuttavia, quando la numerosità campionaria è pari a 100, si ha maggiore difficoltà nell'individuare l'adeguatezza del modello generatore dei dati.

Si verifica ora la capacità del metodo di individuare la mancanza del termine quadratico nel predittore lineare in due situazioni diverse. I dati sono generati in entrambi

TABELLA 3.2: Percentuale di rifiuto del modello parametrico

metodo/numerosità	100	200	300
BAG	26	0	0
RF	26	0	0

i casi dal modello logistico

$$\text{logit}(\pi_i) = 0.5 + 4x_{1i} + 2x_{2i} + 3x_{3i} + x_{6i} + 4x_{1i}^2,$$

con $i = 1, ..., n$. Nel primo caso, indicato con l'espressione "Scenario 1", il modello parametrico è stimato considerando le variabili x_1, x_2, x_3 e x_6, nel secondo caso, detto "Scenario 2", è presente solo la covariata x_6.

Le Figure 3.2 e 3.3 riportano i risultati delle due casistiche. Quando nel modello manca solo il termine quadratico, vi è maggiore difficoltà nell'individuare tale mancanza, soprattutto usando come stimatore non parametrico il *bagging*. Al contrario, nello "Scenario 2" il modello parametrico adattato è sempre, o quasi, rifiutato anche quando la numerosità campionaria è pari a 100.

La potenza nell'individuare problemi nell'adattamento sembra essere elevata per i metodi DRY-V. Tuttavia, alla luce della simulazione effettuata, il metodo *random forest* risulta leggermente preferibile al *bagging*.

3.3 Applicazioni a dati reali

Applicazione all'insieme di dati "Low Birth Weight"

L'insieme di dati *"Low Birth Weight"* contiene informazioni relative a 189 nascite, di cui 59 riguardano neonati sottopeso, identificate nella variabile risposta LOW. I dati sono stati raccolti per uno studio inerente le possibili cause delle nascite sottopeso. In seguito ad un'analisi preliminare si considera un modello contenente le variabili

- età della madre (AGE), variabile quantitativa continua
- peso della madre (LWT), variabile quantitativa continua
- gruppo etnico ($RACE$), variabile qualitativa con 3 livelli, con valore 1 se di carnagione chiara, 2 se scura e 3 altro

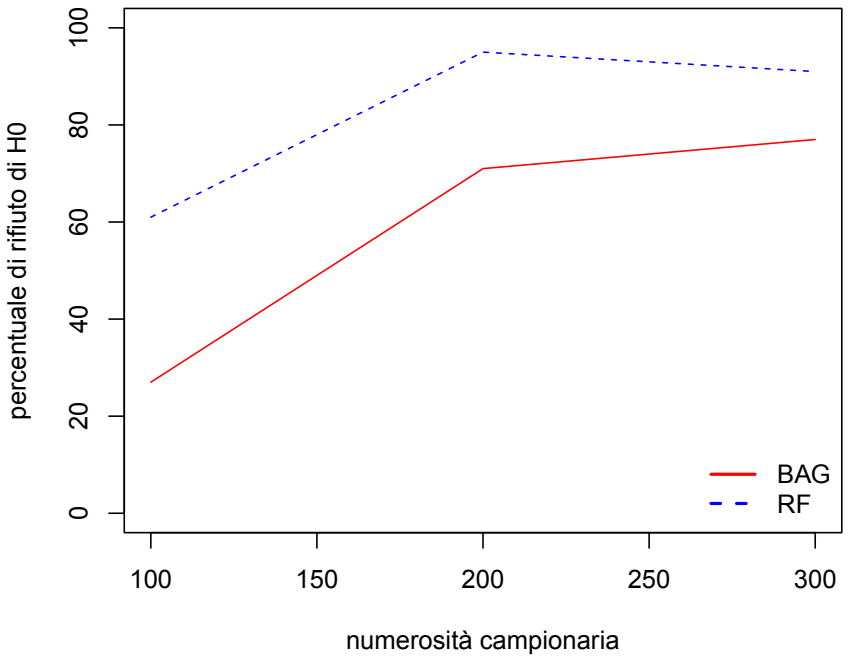

FIGURA 3.2: "Scenario 1". Percentuale di rifiuto del modello parametrico.

- il vizio del fumo (*SMOKE*), variabile dicotomica, con valore 1 se fumatrice, 0 altrimenti.

La variabile *AGE* si indica con x_1, *LWT* con x_2, *RACE* con x_3 se $RACE = 2$ e con x_4 se $RACE = 3$, *SMOKE* con x_5. Si adatta il seguente modello

$$logit(\pi_i) = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i} + \beta_5 x_{5i},$$

con $i = 1, ..., 59$. Nella Tabella 3.3 si riportano le stime del modello.

TABELLA 3.3: Stima dei coefficienti, degli *standard error* e *p-value* del modello adattato.

	stima	*standard error*	statistica z	Pr($>\|z\|$)
β_0	0.332452	1.107672	0.300	0.76407
β_1	-0.022478	0.034170	-0.658	0.51065
β_2	-0.012526	0.006386	-1.961	0.04982
β_3	1.231671	0.517152	2.382	0.01724
β_4	0.943263	0.416232	2.266	0.02344
β_5	1.054439	0.380000	2.775	0.00552

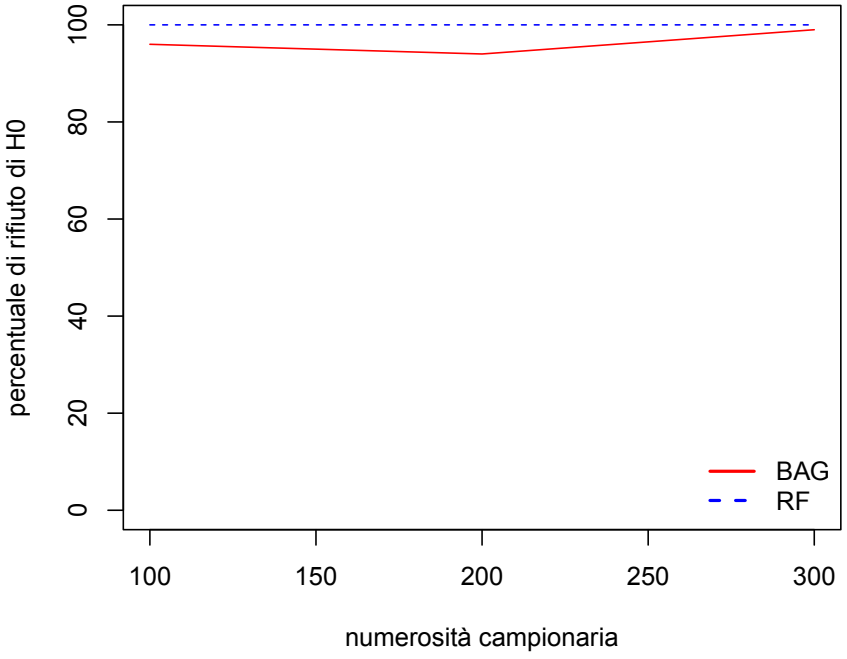

FIGURA 3.3: "Scenario 2". Percentuale di rifiuto del modello parametrico.

La variabile *AGE* viene mantenuta, seppur non significativa, poichè rappresenta una variabile importante per l'analisi. La capacità predittiva del modello risulta discreta osservando la curva ROC, riportata nella Figura 3.4, con un valore dell'AUC pari a 0.684.

Nella Tabella 3.5 si riportano i valori dei seguenti test: le due versioni del test di Hosmer–Lemeshow, il test di le Cessie–van Houwelingen, il test di Stukel e le due versioni del test di Royston. Tutti i test, eccetto quello di Stukel e una delle due versioni

TABELLA 3.4: Valori delle statistiche dei test di valutazione della bontà di adattamento.

test	statistica	df	p-value
$\hat{C}$	10.82	8	0.21
$\hat{H}$	13.54	8	0.09
$\hat{T}$	0.52	8	0.716
$\hat{ST}$	6.60	2	0.036
$\hat{PR}_1$	5.89	*	0.30
$\hat{PR}_2$	8.06	*	0.049

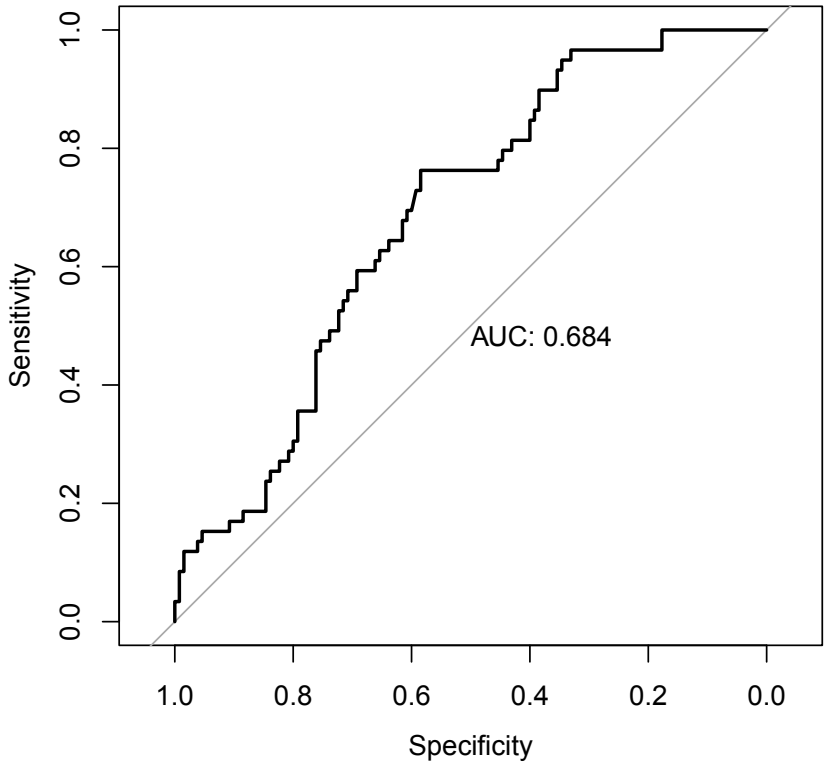

FIGURA 3.4: Grafico della curva ROC.

del test HL, presentano *p-value* superiori a 0.05, indicando un buon adattamento del modello considerato. Si nota il valore molto alto del *p-value* del test basato sui residui standardizzati lisciati e si ottiene un risultato leggermente maggiore di quello riportato nell'articolo Hosmer et al. (1997), questo poichè l'articolo ha calcolato la media e la varianza usando le approssimazioni illustrate nell'Appendice B, mentre qui il *p-value* è stato calcolato utilizzando le stime delle due quantità ottenute mediante simulazione. Il risultato ottenuto è sensato, la distribuzione nulla si approssima discretamente a quella teorica, come è osservabile nel grafico riportato nell'Appendice C.

Si applica sull'insieme di dati il metodo DRY-V, effettuando 300 permutazioni dei dati e in ognuna di essa i dati vengono divisi in due parti, considerando le seguenti casistiche:

- l'insieme di stima contiene il 50% dei dati e l'insieme di verifica ha il restante 50%;
- l'insieme di stima contiene il 90% dei dati l'insieme di verifica ha il restante 10%.

Si indica con l'espressione *glm* il modello parametrico. Sia per i due modelli non parametrici che per il modello di regressione logistica si considerano tutte le variabili esplicative.

Le Tabelle 3.5 e 3.6 riportano le percentuali delle volte in cui un modello ha verosomiglianza predittiva maggiore di un altro modello. Il modello parametrico è chiaramente preferito alle alternative non parametriche, pertanto il modello logistico stimato presenta un buon adattamento. Questo risultato è coerente con la maggior parte dei test di bontà di adattamento.

TABELLA 3.5: 50% training. Il valore in posizione (i,j) riporta la percentuale di volte in cui il i-esimo modello ha verosomiglianza predittiva maggiore dell'j-esimo modello.

	glm	RF	BAG
glm	0	98.7	100
RF	1.3	0	85
BAG	0	15	0

TABELLA 3.6: 90% training. Il valore in posizione (i,j) riporta la percentuale di volte in cui il i-esimo modello ha verosomiglianza predittiva maggiore dell'j-esimo modello.

	glm	RF	BG
glm	0	85	88
RF	15	0	78
BG	17	22	0

Applicazione all'insieme dei dati *"Horseshoe Crab"*

L'insieme dei dati *"Horseshoe Crab"* riporta le misurazioni relative ad un campione di limuli, per valutare come il sesso sia in relazione con alcuni fattori. Dopo aver effettuato un'analisi preliminare, il modello migliore risultante contiene solo la variabile esplicativa quantitativa continua, *width*, che riporta la larghezza dei limuli, indicata con x. Si può ipotizzare, come modello per $(y_1, ..., y_n)$,

$$Y_i \sim \ Bi(1, \pi_i),$$

con $\text{logit}(\pi_i) = \beta_0 + \beta_1 x_i$ e $i = 1, ..., n$.

Nella Tabella 3.7 si riporta l'adattamento del modello logistico, dove entrambi i coefficienti risultano significativi.

TABELLA 3.7: Stima dei coefficienti, degli *standard error* e *p-value* del modello adattato.

	stima	*standard error*	statistica z	$\Pr(>\|z\|)$
β_0	-12.3508	2.6287	-4.698	2.62e-06
β_1	0.4972	0.1017	4.887	1.02e-06

Il modello stimato sembra avere un buon adattamento, come emerge sia dall'indice $AUC = 0.742$, riportato nella Figura 3.5, sia dai risultati dei test. La Tabella 3.8 riporta i valori delle statistiche test e dei relativi *p-value*, i quali sono molto elevati.

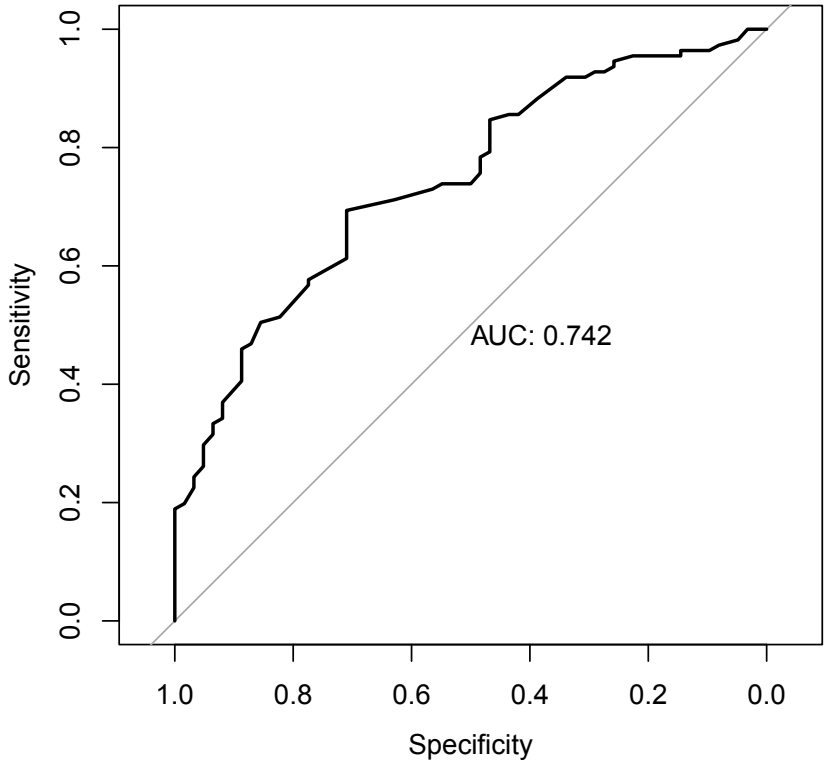

FIGURA 3.5: Grafico della curva ROC.

Per il metodo DRY-V valgono le stesse caratteristiche indicate nel paragrafo 3.3.1. Il modello parametrico risulta preferibile. Si nota che il modello non parametrico selezionato tramite *bagging* è preferibile al modello selezionato tramite *random forest*. I risultati sono riportati nelle Tabelle 3.9 e 3.10. Sia da questo metodo che dai test per la valutazione della bontà di adattamento emerge una valutazione soddisfacente del modello stimato.

TABELLA 3.8: Valori delle statistiche dei test di valutazione della bontà di adattamento

test	statistica	df	p-value
$\hat{C}$	4.39	8	0.82
$\hat{H}$	6.85	8	0.55
$\hat{T}$	0.76	8	0.52
$\hat{ST}$	1.03	2	0.59
$\hat{PR}_1$	4.02	*	0.74
$\hat{PR}_2$	5.19	*	0.37

TABELLA 3.9: 50% training.La (i,j)-esima riga riporta la percentuale di volte che il i-esimo modello ha verosomiglianza predittiva maggiore dell'j-esimo modello

	glm	RF	BAG
glm	0	100.00	99.67
RF	0	0	29.67
BAG	0.33	70.33	0

TABELLA 3.10: 90% training.La (i,j)-esima riga riporta la percentuale di volte che il i-esimo modello ha verosomiglianza predittiva maggiore dell'j-esimo modello

	glm	RF	BAG
glm	0	98.30	97.33
RF	1.70	0	29.00
BAG	2.67	71.00	0

Conclusioni

Lo scopo di questa relazione è presentare gli strumenti per valutare la bontà di adattamento di modelli di regressione per dati binari non raggruppati. Sono stati descritti il test di Hosmer–Lemeshow, il test di le Cessie–van Houwelingen, il test di Stukel e il test di Royston. I test sono stati confrontati attraverso una simulazione sia sotto l'ipotesi nulla che in uno scenario alternativo, e successivamente sono stati applicati a due casi reali. L'analisi svolta riprende studi condotti in Hosmer et al. (1997). Come espresso nel capitolo 3, si può indicare il test di Stukel come il migliore tra i sei test, data la sua buona prestazione in entrambi gli scenari considerati. Inoltre, è stato analizzato il metodo DRY-V (Lu & Yang, 2018), che sfrutta tecniche di *maching learning*. Esso mostra ottime prestazioni come si evince dai risultati della simulazione. Si suggerisce di utilizzare il metodo DRY-V per la valutazione e successivamente confermare il risultato ottenuto applicando il test di Stukel.

Appendice

A Distribuzione chi-quadrato non centrale

La distribuzione chi-quadrato non centrale è una distribuzione di probabilità continua, che si ottiene come somma di quadrati di variabili casuali con distribuzione normale. La distribuzione $\chi^2(k, \lambda)$ si ricava come

$$X^2 = \sum_{i=1}^{k} X_i^2 = X_1^2 + ... + X_k^2,$$

dove $X_1, ..., X_k$ sono variabili aleatorie indipendenti con distribuzioni normali $\mathcal{N}(\mu_1, 1), ...,$ $\mathcal{N}(\mu_k, 1)$. Il parametro λ, definito da

$$\lambda = \sum_{i=k}^{k} \mu_i^2,$$

è detto parametro di non centralità e k sono i gradi di libertà. Se $\lambda = 0$, le variabili X_i sono centrate e quindi si ottiene la distribuzione chi-quadrato centrale χ_k^2. La distribuzione $\chi^2(k, \lambda)$ dipende da λ e non dai valori delle singole medie.

B Calcolo della statistica di le Cessie–van Houwelingen e dei suoi momenti

In questa tesi, per calcolare la statistica test del test di le Cessie–van Houwelingen si riprende la formulazione utilizzata in Hosmer et al. (1997), utilizzando la forma matriciale per ricavare i suoi momenti. Sia W la matrice n x n contenente i pesi w_{ij}, uguali

a $w_{ij} = \prod_{k=1}^{p} u(x_{ik}, x_{jk})$, dove

$$u(x_{ik}, x_{jk}) = \begin{cases} 1 & se|x_{ik} - x_{jk}|/s_k \leq c_u \\ 0 & altrimenti. \end{cases},$$

che rappresentano il peso per definire la distanza tra il soggetto i e j. Il valore c_u è pari a $\frac{1}{2}h_n = \frac{1}{2}(4/n^{1/(2p)})$. In forma matriciale le stime dei residui standardizzati sono

$$\hat{r}_s = \hat{V}^{1/2}\hat{r},$$

dove $\hat{V} = \text{diag}[\hat{\pi}_i(1 - \hat{\pi}_i)]$ è una matrice diagonale n x n e $\hat{r}$ è il vettore dei residui. Di conseguenza i residui standardizzati lisciati sono $\tilde{r}_e = W\hat{r_s}$. La statistica test $\hat{T}$ si esprime come

$$\hat{T} = \frac{1}{n}\tilde{r}_e^T D_r^{-1}\tilde{r}_e, \tag{A.1}$$

dove D_r è una matrice diagonale n x n contenente gli elementi diagonali della matrice WW^T, nonchè le varianze dei residui standardizzati lisciati. Si segnala che nella formula (1) di Hosmer et al. (1997) manca la costante $\frac{1}{n}$.

Sia $A_r = (I-M)^T Q_r (I-M)$, con I matrice diagonale n x n, $M = VX(X^TVX)^{-1}X^T$ e $Q_r = V^{-1/2}(W^T D_r^{-1} W)V^{-1/2}$. I momenti della statistica $\hat{T}$ sono

$$E(\hat{T}) = \frac{1}{n}tr(A_r\hat{V})$$

e

$$\hat{var}(\hat{T}) \cong 2\left(\frac{2}{3}\right)^p \frac{tr(WW^T)}{n^2},$$

dove la traccia di una matrice è la somma degli elementi sulla diagonale. Per l'implementazione di questo test la media e la varianza sono state stimate mediante simulazione e la statistica test è stata calcolata usando la formula (A.1).

C Codice R

Figura 2.1

```
#relazione tra p-value del test HL tradizionale e p-value del test modificato
library(DescTools)
pvalue = function(n, N, G){
  ordinata = c(rep(0,N))     # pvalue modificato
```

```
  ascissa = c(rep(0,N))    # pvalue tradizionale
  for (i in 1:N) {
    y = rbinom(n, size = 1, prob = 0.5)
    x1 = rnorm(n)
    x2 = rnorm(n)
    mod = glm(y ~ x1+x2, family = binomial)
    C = as.numeric(DescTools::HosmerLemeshowTest(fitted(mod),y, ngr = G)$C[1])
    epsilon0 = sqrt((qchisq(p = 0.95, df = G-2, ncp = 0)-G-2)/10^6)
    ordinata[i] = 1 - pchisq(q = C, df = G-2, ncp = (epsilon0^2)*n)
    ascissa[i] = 1 - pchisq(q = C, df =G-2)
  }
  return (cbind(ascissa,ordinata))
}
grafico = pvalue(10000,200,10)
grafico1 = pvalue(50000,200,10)
grafico2 = pvalue(100000,200,10)
grafico3 = pvalue(500000,200,10)
matrice_ordinata = grafico[order(grafico[,1]),]
plot(matrice_ordinata, type = "l", ylab = "p-value test modificato",
   xlab = "p-value test tradizionale", xlim = c(0,1), ylim = c(0,1), lwt = 2 )
matrice_ordinata1 = grafico1[order(grafico1[,1]),]
lines(matrice_ordinata1, type = "l", col = "red", lty = 2, lwd= 2)
matrice_ordinata2 = grafico2[order(grafico2[,1]),]
lines(matrice_ordinata2, type = "l", col = "green", lty = 3, lwd = 2)
matrice_ordinata3 = grafico3[order(grafico3[,1]),]
lines(matrice_ordinata3, type = "l", col = "blue", lty = 4, lwd = 2)
legend("bottomright",
   legend = c("n = 10000",  "n = 50000", "n = 100000","n = 500000"),
   col = c("black", "red", "green", "blue"), lty = c(1,2,3,4), lwd = 2,
   bty  = "n")
```

Simulazione

```
set.seed(123)
library(DescTools)
library(LogisticDx)
```

```
# Royston
royston1 = function(y, mod){
  pihat = fitted(mod)
  n = length(pihat)
  m = c(rep(0, n))
  pihat_s = sort(pihat, decreasing = TRUE)    # valori stimati ordinati
  y_s = as.numeric(names(pihat_s)) # indici ordinati delle risposte corrispondenti
  for (i in 1:n) {
    q = c(rep(0,i))
    for (j in 1:i) {
      q[j] = (y[y_s[j]] - pihat_s[j])
    }
    somma = - sum(q)
    m[i] = abs(somma)
  }
  return (max(m))
}

royston2 = function(y, mod){
  pihat = fitted(mod)
  n = length(pihat)
  m = c(rep(0, n))
  pihat_s = sort(pihat, decreasing = TRUE)    # valori stimati ordinati
  y_s = as.numeric(names(pihat_s)) # indici ordinati delle risposte corrispondenti
  for (i in 1:n) {
    q = c(rep(0,i))
    for (j in 1:i) {
      q[j] = (y[y_s[j]] - pihat_s[j])
    }
    m[i] = -sum(q)
  }
  differenza = c(rep(0,n/2))
  for (i in 1:(n/2)){
    differenza[i] =abs(m[i]-m[n-i])
  }
```

```
  return (max(differenza))
}

w = function(x, i, j, c_u, s, p) {
  U = rep(0,p)
  if (p > 1) {
    for (k in 1:p) {
      u = abs(x[i,k] - x[j,k]) / s[k]
      if ( u <= c_u ) {
        U[k] = 1}
      else {
        U[k] = 0}
    }
    return (prod(U))}
  else {
    u = abs(x[i] - x[j]) / s
    U[ u <= c_u ] = 1
    return(U)}
}


# matrice dei pesi
matrice_pesi = function(X){
  n = nrow(X)
  p = ncol(X)
  W = matrix ( 0, ncol = n, nrow = n)
  c_u = 0.5 * (4 / (n^(1/(2*p))))
  s = sqrt(diag(var(X)))
  for (i in 1:n) {
    for (j in 1:n) {
      W[i,j] = w(X, i, j, c_u, s, p)
    }
  }
  return(W)
}
```

```
leCessie = function(mod,y,W) {
  X = model.matrix(mod)
  n = nrow(X)
  p = ncol(X)-1
  X = X[,-1]
  pihat = fitted(mod)
  e = y - pihat
  V = diag(pihat * (1 - pihat))
  V_m05 = diag( (pihat * (1 - pihat))^(-0.5) )
  r_s = V_m05 %*% e      # residui standardizzati
  r_tilde = W %*% r_s
  D_r = diag( diag(W %*% t(W)) )
  T = (1/n) * t(r_tilde) %*% solve(D_r) %*% r_tilde
  return (T)
}

media_emp = function(n,N) {
  quant = c(rep(0,N))
  x1 = as.matrix(runif(n, min = -6, max = 6))
  W = matrice_pesi(x1)
  for (i in 1:N) {
    y = runif(n, min = 0, max = 1)
    y1 = ifelse( y<= exp(1.40+0.85*x1)/(1+exp(1.40+0.85*x1) ), 1, 0)
    mod = glm(y1 ~  x1, family = binomial)
    quant[i] = leCessie(mod,y1,W)
  }
  media = mean(quant)
  varianza = var(quant)
  assign("punto_criticoT", quant, envir = .GlobalEnv)
  gdl = 2*(media)^2/varianza
  c = varianza/2/media
  assign("c", c, envir = .GlobalEnv)
  assign("gdl", gdl, envir = .GlobalEnv)
  return(c(media, varianza))
```

```
}

# ipotesi nulla
ipotesi_nulla = function(n,N) {
  quantH = pH = c(rep(0,N))
  quantC = pC = c(rep(0,N))
  quantST = pST = c(rep(0,N))
  quantPR1 = pPR1 = c(rep(0,N))
  quantPR2 = pPR2 = c(rep(0,N))
  quantT = pT = c(rep(0,N))
  # per il test di le Cessie
  valori_empirici = media_emp(n, N)
  media_empirica = valori_empirici[1]
  varianza_empirica = valori_empirici[2]
  x1 = runif(n, min = -6, max = 6)
  for (i in 1:N) {
    cat("\ni: ",i)
    y = runif(n, min = 0, max = 1)
    y1 = ifelse( y<= exp(1.40+0.85*x1)/(1+exp(1.40+0.85*x1) ), 1, 0)
    mod = glm(y1 ~ x1, family = binomial)
    # statistica H
    quantH[i] = as.numeric(HosmerLemeshowTest(fitted(mod), y1)$H[1])
    G = as.numeric(HosmerLemeshowTest(fitted(mod), y1)$H[2])
    if (quantH[i] >qchisq(0.95, df = G)){
      pH[i] = 1
    }
    # statistica C
    quantC[i] = as.numeric(HosmerLemeshowTest(fitted(mod), y1)$C[1])
    G = as.numeric(HosmerLemeshowTest(fitted(mod), y1)$C[2])
    if (quantC[i] >qchisq(0.95, df = G)){
      pC[i] = 1
    }
    # statistica ST
    quantST[i] = as.numeric(LogisticDx::stukel(mod)[1])
    if (quantST[i] >qchisq(0.95, df = 2)){
      pST[i] = 1
```

```
  }
  # statistica PR1
  quantPR1[i] = royston1(y1, mod)
  # calcolo il pvalue
  quant = log(quantPR1[i])-0.5236*log(n)
  # questa quantità si approssima ad una normale di media mu1 e sd sigma1
  m = sum(y1)
  L = log(m/(n-m))
  mu1 = -1.0605-0.12684*(L^2)-0.010437*(L^3)
  sigma1 = 0.3130+0.004261*L
  if ( quant >= qnorm(0.95, mean = mu1, sd = sigma1) ) {
    pPR1[i] = 1
  }
  # statistica PR2
  quantPR2[i] = royston2(y1, mod)
  # calcolo il pvalue
  m = sum(y1)
  L = log(m/(n-m))
  quant1 = log(quantPR2[i])-0.5236*log(n)
  mu2 = -1.1196-0.12093*(L^2)-0.009744*(L^3)
  sigma2 = 0.3269+0.00654*L
  if ( quant1 >= qnorm(0.95, mean = mu2, sd = sigma2) ) {
    pPR2[i] = 1
  }
  # statistica T
  gdl = 2*(media_empirica)^2/varianza_empirica
  c = varianza_empirica/2/media_empirica
  if (punto_criticoT[i]/c > qchisq(0.95, df = gdl)) {
    pT[i] = 1 }
}
assign("quantile_osservato_H", quantile(quantH, p = 0.95), envir = .GlobalEnv)
assign("quantile_osservato_C", quantile(quantC, p = 0.95), envir = .GlobalEnv)
assign("quantile_osservato_ST",
       quantile(quantST, p = 0.95), envir = .GlobalEnv)
assign("quantile_osservato_PR1",
        quantile(quantPR1, p = 0.95), envir = .GlobalEnv)
```

```
  assign("quantile_osservato_PR2",
         quantile(quantPR2, p = 0.95), envir = .GlobalEnv)
  assign("quantile_osservato_T",
         quantile(punto_criticoT, p = 0.95), envir = .GlobalEnv)

  # calcolo alfa effettivo
  alfaH = mean(pH)*100
  alfaC = mean(pC)*100
  alfaST = mean(pST)*100
  alfaPR1 = mean(pPR1)*100
  alfaPR2 = mean(pPR2)*100
  alfaPR1 = alfaPR2 = 0
  alfaT = mean(pT)*100
  alfa_effettivi = cbind(alfaH, alfaC, alfaST, alfaPR1, alfaPR2, alfaT)
  colnames(alfa_effettivi) = c("H", "C", "ST", "PR1", "PR2", "T")
  return( alfa_effettivi)
}

potenza = function(n, N, f) {
  potenzaH = quantH_1 = c(rep(0,N))
  potenzaC = quantC_1 = c(rep(0,N))
  potenzaST = quantST_1 = c(rep(0,N))
  potenzaPR1 = quantPR1_1 = c(rep(0,N))
  potenzaPR2 = quantPR2_1 = c(rep(0,N))
  potenzaT = quantT_1 = c(rep(0,N))
  quantH_0 = quantile_osservato_H
  quantC_0 = quantile_osservato_C
  quantST_0 = quantile_osservato_ST
  quantPR1_0 = quantile_osservato_PR1
  quantPR2_0 = quantile_osservato_PR2
  quantT_0 = quantile_osservato_T
  # modello alternativo ( contiene termine quadratico)
  x1 = runif(n, min = -6, max = 6)
  W = matrice_pesi(cbind(x1,x1^2))
  for (i in 1:N) {
    y = runif(n, min = 0, max = 1)
```

```
  y1 = ifelse( y<= exp(1.40+0.85*x1-0.11*x1^2)/(1+exp(1.40+0.85*x1-0.11*x1^2)), 1,0)
  mod1 = glm(y1 ~ x1, family = binomial)
  # statistica H
  quantH_1[i] = as.numeric(HosmerLemeshowTest(fitted(mod1), y1)$H[1])
  if (quantH_1[i] >= quantH_0  ) {
    potenzaH[i] = 1
  }
  # statistica C
  quantC_1[i] = as.numeric(HosmerLemeshowTest(fitted(mod1), y1)$C[1])
  if (quantC_1[i] >= quantC_0  ) {
    potenzaC[i] = 1
  }
  # statistica ST
  quantST_1[i] = as.numeric( LogisticDx::stukel(mod1)[1])
  if (quantST_1[i] >= quantST_0  ) {
    potenzaST[i] = 1
  }
  # statistica PR1
  quantPR1_1[i] = royston1(y1, mod1)
  if (quantPR1_1[i] >= quantPR1_0  ) {
    potenzaPR1[i] = 1
  }
  # statistica PR2
  quantPR2_1[i] = royston2(y1, mod1)
  if (quantPR2_1[i] >= quantPR2_0  ) {
    potenzaPR2[i] = 1
  }
  # statistica T
  quantT_1[i] = leCessie( mod1, y1,W)
  if (quantT_1[i] >= quantT_0  ) {
    potenzaT[i] = 1
  }

}
alfaH = mean(potenzaH)*100
alfaC = mean(potenzaC)*100
```

```
  alfaST = mean(potenzaST)*100
  alfaPR1 = mean(potenzaPR1)*100
  alfaPR2 = mean(potenzaPR2)*100
  alfaT = mean(potenzaT)*100
  alfa_effettivi = cbind(alfaH, alfaC, alfaST, alfaPR1, alfaPR2, alfaT)
  colnames(alfa_effettivi) = c("H", "C", "ST", "PR1", "PR2", "T")
  return( alfa_effettivi)
}

x1 = ipotesi_nulla(100, 1000)
x1_potenza = potenza(100, 1000)
x2 = ipotesi_nulla(200, 1000)
x2_potenza = potenza(200, 1000)
x3 = ipotesi_nulla(300, 1000)
x3_potenza = potenza(300, 1000)
x4 = ipotesi_nulla(400, 1000)
x4_potenza = potenza(400, 1000)
x5 = ipotesi_nulla(500, 1000)
x5_potenza = potenza(500, 1000)

# grafico
X = rbind(x1_potenza, x2_potenza, x3_potenza, x4_potenza, x5_potenza)
y = c(100, 200, 300, 400, 500)
plot(y, X[,1], xlab = "numerosità campionaria",
     ylab = "percentuale di rifiuto di H0",
     ylim = c(0,100), xlim = c(100,500), type = "l", col = "red")
lines(y, X[,2], col = "green", lty = 2)
lines(y, X[,3], col = "blue", lty = 3)
lines(y, X[,4], col = "violet", lty = 4)
lines(y, X[,5], col = "yellow", lty = 5)
lines(y, X[,6] , col = "turquoise", lty = 6)
legend("bottomright", legend = c("H",  "C", "ST","PR1", "PR2", "T"),
       col = c("red", "green", "blue", "violet", "yellow", "turquoise"),
       lty = c(1,2,3,4,5,6), lwd = 2, bty  = "n")

# grafico per valutare la distribuzione empirica nulla della statistica T
```

```
x <- seq(0, 20, length.out = 100)
y <- dchisq(x, df = gdl)
data <- data.frame(punto_criticoT/c)
ggplot(data, aes(x = punto_criticoT/c)) +
  geom_density(fill = "hotpink4", col = "hotpink4") +
  geom_line(data = data.frame(x, y), aes(x, y), color = "blue", size = 1) +
  labs(x = "Statistiche", y = "Densità") +
  ggtitle("Distribuzione delle statistiche T generate attraverso
          simulazione di Monte Carlo") +
  theme_minimal()
```

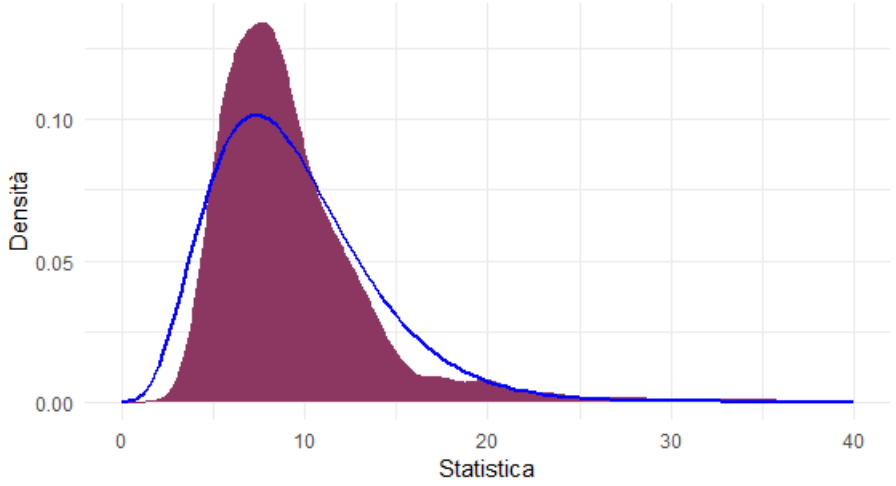

FIGURA .1: Grafico della distribuzione nulla empirica della statistica T

```
# SIMULAZIONE METODO DRY-V
set.seed(123)
library(randomForest)
library(tidyverse)
library(MASS)

CV_ipotesi = function(x, p, rf){
  # calcolo CV dello stimatore parametrico
  pred_p = predict(p, newdata = x ,type = "response")
  CV_p = t(x$y1) %*% log(pred_p) + t(1-x$y1) %*% log(1-pred_p)
  # calcolo CV dello stimatore non parametrico
  pred_np = predict(rf, newdata = x, type = "prob")
  pred_np1 = ifelse(pred_np[,2] %in% c(0, 1), c(0.001,0.999),pred_np[,2])
```

```
  CV_np = t(x$y1) %*% log(pred_np1) + t(1-x$y1) %*% log(1-pred_np1)
  if ( CV_p >= CV_np )
  {return (1)}
  else
  {return (0)}
}

# bagging
bag_ipotesi = function(data,n) {
  risultato = rep(0,n)
  for (i in 1:n) {
    # permutazione degli ordini casualmente
    idx = sample(1:nrow(data), 0.75*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    if ( length(unique(train[,1]))==1 ) {
      next
    }
    p = glm(y1 ~  x1 + x2 + x3 + x4 + x5 + x6 , family = binomial, data = train)
    np = randomForest( as.factor(y1) ~ x1 + x2 + x3 + x4 + x5 + x6 , data = train,
                       replace = TRUE, mtry = 6)
    risultato[i] = CV_ipotesi(test, p, np)
  }
  if ( sum(risultato) >= (n/2) ) {
    soluzione = 1
  }
  else {
    soluzione = 0
  }
  return (soluzione)
}

# random forest
rf_ipotesi = function(data,n) {
  risultato = rep(0,n)
  for (i in 1:n) {
```

```
    idx = sample(1:nrow(data), 0.75*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    if ( length(unique(train[,1]))==1 ) {
      next
    }
    p = glm(y1 ~  x1 + x2 + x3 + x4 + x5 + x6 , family = binomial, data = train)
    np = randomForest( as.factor(y1) ~ x1 + x2 + x3 + x4 + x5 + x6 ,
                       data = train, replace = FALSE)
    risultato[i] = CV_ipotesi(test, p, np)
  }
  if ( sum(risultato) >= (n/2) ) {
    soluzione = 1
  }
  else {
    soluzione = 0
  }
  return (soluzione)
}

# simuazione ipotesi nulla sia bagging che random forest
simulazione_ipotesi = function(n, N){
  perc_bag = c(rep(0,N))
  perc_rf = c(rep(0,N))
  x0 = c(rep(0.5,n))
  x1 = rnorm(n, mean = 0, sd = 1)
  x2 = rnorm(n, mean = 0, sd = 1)
  x3 = rnorm(n, mean = 0, sd = 1)
  x4 = rnorm(n, mean = 0, sd = 1)
  x = mvrnorm(n, mu = c(0, 0), Sigma = matrix(c(1, 0.5, 0.5, 1), ncol = 2))
  x5 = x[,1]
  x6 = x[,2]
  for (i in 1:N){
    z = runif(n, min = 0, max = 1)
    g1 = x0 + 4*x1 + 2*x2 + x3 - 5*x4 -x5 +3*x6
    y1 = ifelse( z<= exp(g1)/(1+exp(g1) ), 1, 0)
```

```
    data = as.data.frame(cbind(y1, x1, x2, x3, x4, x5, x6))
    # bagging
    perc_bag[i] = bag_ipotesi(data,n)
    # random forest
    perc_rf[i] = rf_ipotesi(data,n)
  }
  alfa_bag = (1-mean(perc_bag))*100
  alfa_rf = (1-mean(perc_rf))*100
  alfa = cbind(alfa_bag, alfa_rf)
  colnames(alfa) = c("bag","rf")
  return (alfa)
}

x1_ipotesi = simulazione_ipotesi(100, 100)
x2_ipotesi = simulazione_ipotesi(200, 100)
x3_ipotesi = simulazione_ipotesi(300, 100)

# scenario 1
set.seed(123)
library(MASS)
library(randomForest)
library(tidyverse)
CV_large = function(x, p, rf){
  pred_p = predict(p, newdata = x[,c(1,2,3,4,7)] ,type = "response")
  CV_p = t(x$y1) %*% log(pred_p) + t(1-x$y1) %*% log(1-pred_p)
  pred_np = predict(rf, newdata = x, type = "prob")
  pred_np1 = ifelse(pred_np[,2] %in% c(0, 1), c(0.001,0.999),pred_np[,2])
  CV_np =t(x$y1) %*% log(pred_np1) + t(1-x$y1) %*% log(1-pred_np1)
  if ( CV_p>=CV_np)
  {return (1)}
  else
  {return (0)}
}

# bagging
potenza_bag_large = function(data,n) {
```

```
  risultato = rep(0,n)
  for (i in 1:n) {
    idx = sample(1:nrow(data), 0.75*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    if ( length(unique(train[,1]))==1 ) {
      next
    }
    p = glm(y1 ~  x1 + x2 + x3 + x6  , family = binomial, data = train)
    np = randomForest( as.factor(y1) ~ x1 + x2 + x3 + x4 + x5 + x6 ,
                       data = train, replace = TRUE, mtry = 6)
    risultato[i] = CV_large(test, p, np)
  }
  if ( sum(risultato) >= (n/2) ) {
    soluzione = 1
  }
  else {
    soluzione = 0
  }
  return (soluzione)
}

# random forest
potenza_rf_large = function(data,n) {
  risultato = rep(0,n)
  for (i in 1:n) {
    idx = sample(1:nrow(data), 0.75*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    if ( length(unique(train[,1]))==1 ) {
      next
    }
    p = glm(y1 ~  x1 + x2 + x3 + x6  , family = binomial, data = train)
    np = randomForest( as.factor(y1) ~ x1 + x2 + x3 + x4 + x5 + x6  , data = train,
                       replace = FALSE)
    risultato[i] = CV_large(test, p, np)
```

```
  }
  if ( sum(risultato) >= (n/2) ) {
    soluzione = 1
  }
  else {
    soluzione = 0
  }
  return (soluzione)
}

potenza_large = function(n, N){
  perc_bag = c(rep(0,N))
  perc_rf = c(rep(0,N))
  x0 = c(rep(0.5,n))
  x1 = rnorm(n, mean = 0, sd = 1)
  x2 = rnorm(n, mean = 0, sd = 1)
  x3 = rnorm(n, mean = 0, sd = 1)
  x4 =  rnorm(n, mean = 0, sd = 1)
  x = mvrnorm(n, mu = c(0, 0), Sigma = matrix(c(1, 0.5, 0.5, 1), ncol = 2))
  x5 = x[,1]
  x6 = x[,2]
  for (i in 1:N){
    cat("\ni: ",i)
    z = runif(n, min = 0, max = 1)
    g1 = x0 + 4*x1 + 2*x2 + 3*x3 + x6 + 4*(x1^2)
    y1 = ifelse( z<= exp(g1)/(1+exp(g1) ), 1, 0)
    data = as.data.frame(cbind(y1, x1, x2, x3, x4, x5, x6))
    # bagging
    perc_bag[i] = potenza_bag_large(data,n)
    # random forest
    perc_rf[i] = potenza_rf_large(data,n)
  }
  alfa_bag = (1-mean(perc_bag))*100
  alfa_rf = (1-mean(perc_rf))*100
  alfa = cbind(alfa_bag, alfa_rf)
  colnames(alfa) = c("bag","rf")
```

```
  return (alfa)
}

x1_large = potenza_large(100, 100)
x2_large = potenza_large(200, 100)
x3_large = potenza_large(300, 100)

# grafico
X = rbind(x1_large, x2_large, x3_large)
y = c(100, 200, 300)
plot(y, X[,1], xlim = c(100,300), ylim = c(0,100),
     xlab = "numerosità campionaria",
     ylab = "percentuale di rifiuto di H0", type = "l", col = "red")
lines(y, X[,2], col = "blue", lty = 2)
legend("bottomright", legend = c("BAG", "RF"),
       col = c( "red", "blue"), lty = c(1,2), lwd = 2, bty  = "n")


# scenario 2
set.seed(123)
library(MASS)
library(randomForest)
library(tidyverse)
CV_small = function(x, p, rf){
  pred_p = predict(p, newdata = x[,c(1,7)] ,type = "response")
  CV_p = t(x$y1) %*% log(pred_p) + t(1-x$y1) %*% log(1-pred_p)
  pred_np = predict(rf, newdata = x, type = "prob")
  pred_np1 = ifelse(pred_np[,2] %in% c(0, 1), c(0.001,0.999),pred_np[,2])
  CV_np =t(x$y1) %*% log(pred_np1) + t(1-x$y1) %*% log(1-pred_np1)
  if ( CV_p>=CV_np)
  {return (1)}
  else
  {return (0)}
}

# bagging
```

```
potenza_bag_small = function(data,n) {
  risultato = rep(0,n)
  for (i in 1:n) {
    idx = sample(1:nrow(data), 0.75*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    if ( length(unique(train[,1]))==1 ) {
      next
    }
    p = glm(y1 ~ x6 , family = binomial, data = train)
    np = randomForest( as.factor(y1) ~ x1 + x2 + x3 + x4 + x5 + x6 , data =train,
                       replace = TRUE, mtry = 6)
    risultato[i] = CV_small(test, p, np)
  }
  if ( sum(risultato) >= (n/2) ) {
    soluzione = 1
  }
  else {
    soluzione = 0
  }
  return (soluzione)
}

# random forest
potenza_rf_small = function(data,n) {
  risultato = rep(0,n)
  for (i in 1:n) {
    idx = sample(1:nrow(data), 0.75*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    if ( length(unique(train[,1]))==1 ) {
      next
    }
    p = glm(y1 ~ x6 , family = binomial, data = train)
    np = randomForest( as.factor(y1) ~ x1 + x2 + x3 + x4 + x5 + x6, data = train,
                       replace = FALSE)
```

```
    risultato[i] = CV_small(test, p, np)
  }
  if ( sum(risultato) >= (n/2) ) {
    soluzione = 1
  }
  else {
    soluzione = 0
  }
  return (soluzione)
}

potenza_small = function(n, N){
  perc_bag = c(rep(0,N))
  perc_rf = c(rep(0,N))
  x0 = c(rep(0.5,n))
  x1 = rnorm(n, mean = 0, sd = 1)
  x2 = rnorm(n, mean = 0, sd = 1)
  x3 = rnorm(n, mean = 0, sd = 1)
  x4 = rnorm(n, mean = 0, sd = 1)
  x = mvrnorm(n, mu = c(0, 0), Sigma = matrix(c(1, 0.5, 0.5, 1), ncol = 2))
  x5 = x[,1]
  x6 = x[,2]
  for (i in 1:N){
    z = runif(n, min = 0, max = 1)
    g1 = x0 + 4*x1 + 2*x2 + 3*x3 + x6 + 4*(x1^2)
    y1 = ifelse( z<= exp(g1)/(1+exp(g1) ), 1, 0)
    data = as.data.frame(cbind(y1, x1, x2, x3, x4, x5, x6))
    # bagging
    perc_bag[i] = potenza_bag_small(data,n)
    # random forest
    perc_rf[i] = potenza_rf_small(data,n)
  }
  alfa_bag = (1-mean(perc_bag))*100
  alfa_rf = (1-mean(perc_rf))*100
  alfa = cbind(alfa_bag, alfa_rf)
  colnames(alfa) = c("bag","rf")
```

```
  return (alfa)
}

x1_small = potenza_small(100, 100)
x2_small = potenza_small(200, 100)
x3_small = potenza_small(300, 100)

# grafico
X = rbind(x1_small, x2_small, x3_small)
y = c(100, 200, 300)
plot(y, X[,1], xlim = c(100,300), ylim = c(0,100), xlab = "numerosità campionaria"
     ylab = "percentuale di rifiuto di H0", type = "l", col = "red")
lines(y, X[,2], col = "blue", lty = 2)
legend("bottomright", legend = c("BAG", "RF"),
       col = c( "red", "blue"), lty = c(1,2), lwd = 2, bty  = "n")
```

Applicazioni sui dati reali

```
# dataset ‘‘Low Birth Weight
# i dati sono reperibili usando il link
# http://danstan.com/students/lowbirthweightdata.txt
dati = read.table("LOWBWT.txt", header = TRUE)
attach(dati)
table(LOW)
RACE = as.factor(RACE)
LBW.glm = glm(LOW ~ AGE + LWT + RACE + SMOKE, family = binomial)
summary(LBW.glm)

# curva ROC
library(pROC)
plot(roc(LOW, fitted(LBW.glm)), print.auc = TRUE)

# test di Hosmer--Lemeshow
DescTools::HosmerLemeshowTest(fitted(LBW.glm),LOW, ngr = 10)
```

```
# test di Stukel
library(LogisticDx)
LogisticDX::stukel(LBW.glm)

# test di Royston
royston1 = function(y, mod){
  pihat = fitted(mod)
  n = length(pihat)
  m = c(rep(0, n))
  pihat_s = sort(pihat, decreasing = TRUE)
  y_s = as.numeric(names(pihat_s))
  for (i in 1:n) {
    q = c(rep(0,i))
    for (j in 1:i) {
      q[j] = (y[y_s[j]] - pihat_s[j])
    }
    somma = - sum(q)
    m[i] = abs(somma)
  }
  return (max(m))
}

PR1 = royston1(LOW, LBW.glm)
PR1
n = length(LOW)
PR1n = log(PR1)-0.5236*log(n)
m = sum(LOW)
L = log(m/(n-m))
mu1 = -1.0605-0.12684*(L^2)-0.010437*(L^3)
sigma1 = 0.3130+0.004261*L
1-pnorm(q = (PR1n-mu1)/sigma1)

royston2 = function(y, mod){
  pihat = fitted(mod)
  n = length(pihat)
  m = c(rep(0, n))
```

```
  pihat_s = sort(pihat, decreasing = TRUE)
  y_s = as.numeric(names(pihat_s))
  for (i in 1:(n)) {
    q = c(rep(0,i))
    for (j in 1:i) {
      q[j] = (y[y_s[j]] - pihat_s[j])
    }
    m[i] = -sum(q)
  }
  differenza = c(rep(0,n/2))
  for (i in 1:(n/2)){
    differenza[i] =abs(m[i]-m[n-i])
  }
  return (max(differenza))
}

PR2 = royston2(LOW, LBW.glm)
PR2n = log(PR2)-0.5236*log(n)
mu2 = -1.1196-0.12093*(L^2)-0.009744*(L^3)
sigma2 = 0.3269+0.00654*L
1-pnorm(q = (PR2n-mu2)/sigma2)

# test di le Cessie--van Houwelingen
X = model.matrix(LBW.glm)[,-1]
n = nrow(X)
p = ncol(X)
pihat = fitted(LBW.glm)
y = LOW
e = y - pihat
V = diag(pihat * (1 - pihat))
V_m05 = diag( (pihat * (1 - pihat))^(-0.5) )
r_s = V_m05 %*% e
p=2
c_u = 0.5 * (4 / (n^(1/(2*p))))
s = sqrt(diag(var(X[,c(1,2)])))
w = function(x, i, j) {
```

```
  U = c(rep(0,p))
  for (k in 1:p) {
    u = abs(x[i,k] - x[j,k]) / s[k]
    if ( u <= c_u ) {
      U[k] = 1}
    else {
      U[k] = 0}
  }
  prodotto = prod(U)
  return (prodotto)
}
W = matrix ( 0, ncol = n, nrow = n)
for (i in 1:n) {
  for (j in 1:n) {
    W[i,j] = w(X, i, j)
  }
}
r_tilde = W %*% r_s
D_r = diag( diag(W %*% t(W)) )
T = (1/n)*t(r_tilde) %*% solve(D_r) %*% r_tilde

# stima della media e varianza mediante simulazione
matrice_pesi = function(X){
  n = nrow(X)
  p = ncol(X)
  W = matrix ( 0, ncol = n, nrow = n)
  c_u = 0.5 * (4 / (n^(1/(2*p))))
  s = sqrt(diag(var(X)))
  for (i in 1:n) {
    for (j in 1:n) {
      W[i,j] = w(X, i, j, c_u, s, p)}
  }
  return(W)
}

leCessie = function(mod,y,W) {
```

```
  X = model.matrix(mod)
  n = nrow(X)
  p = ncol(X)-1
  X = X[,-1]
  pihat = fitted(mod)
  e = y - pihat
  V = diag(pihat * (1 - pihat))
  V_m05 = diag( (pihat * (1 - pihat))^(-0.5) )
  r_s = V_m05 %*% e      # residui standardizzati
  r_tilde = W %*% r_s
  D_r = diag( diag(W %*% t(W)) )
  T = (1/n) * t(r_tilde) %*% solve(D_r) %*% r_tilde
  return (T)
}

media_emp = function(n,N) {
  quant = c(rep(0,N))
  x1 = as.matrix(runif(n, min = -6, max = 6))
  W = matrice_pesi(x1)
  for (i in 1:N) {
    cat("\ni: ",i)
    y = runif(n, min = 0, max = 1)
    y1 = ifelse( y<= exp(1.40+0.85*x1)/(1+exp(1.40+0.85*x1) ), 1, 0)
    mod = glm(y1 ~  x1, family = binomial)
    quant[i] = leCessie(mod,y1,W)
  }
  media = mean(quant)
  varianza = var(quant)
  assign("punto_criticoT", quant, envir = .GlobalEnv)
  gdl = 2*(media)^2/varianza
  c = varianza/2/media
  assign("c", c, envir = .GlobalEnv)
  assign("gdl", gdl, envir = .GlobalEnv)
  return(c(media, varianza))
}
valori = media_emp(400,1000)
```

```
mT = valori[1]    # stima della media
varT = valori[2]  # stima della varianza
b = 2*mT/varT
v = 2*(mT^2)/varT
# pvalue
1 - pchisq(q = b*T/n, df = v)

#######################################################
library(randomForest)
library(tidymodels)
library(glmtoolbox)
set.seed(1234)

confronto_rf = function(x, mod, rf){
  pred_p = predict(mod, newdata = x ,type = "response")
  CV_p = t(x$LOW) %*% log(pred_p) + t(1-x$LOW) %*% log(1-pred_p)
  pred_np = predict(rf, newdata = x, type = "prob")
  pred_np1 = ifelse(pred_np[,2] ==0, 0.001, pred_np[,2])
  CV_np =t(x$LOW) %*% log(pred_np1) + t(1-x$LOW) %*% log(1-pred_np1)
  if ( CV_p>=CV_np)
  {return (1)}
  else
  {return (0)}
}

metodo_rf = function(data, alfa) {
  n = 300
  risultato = rep(0,n)
  for (i in 1:n) {
    idx = sample(1:nrow(data), alfa*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    p = glm(LOW ~ AGE + LWT + RACE + SMOKE, family = binomial, data = train)
    np = randomForest( as.factor(LOW) ~ AGE + LWT + RACE + SMOKE, data = train,
         replace = FALSE)
    risultato[i] = confronto_rf(test, p, np)
```

```
  }
  percentuale = sum(risultato)/n*100
  return (percentuale)
}
metodo_rf(dati, 0.5)
metodo_rf(dati, 0.9)

confronto_bg = function(x, mod, bg){
  pred_p = predict(mod, newdata = x ,type = "response")
  CV_p = t(x$LOW) %*% log(pred_p) + t(1-x$LOW) %*% log(1-pred_p)
  pred_np = predict(bg, newdata = x, type = "prob")
  pred_np1 = ifelse(pred_np[,2] ==0, 0.001, pred_np[,2])
  CV_np =t(x$LOW) %*% log(pred_np1) + t(1-x$LOW) %*% log(1-pred_np1)
  if ( CV_p>=CV_np)
  {return (1)}
  else
  {return (0)}
}

metodo_bg = function(data, alfa) {
  n = 300
  risultato = rep(0,n)
  for (i in 1:n) {
    idx = sample(1:nrow(data), alfa*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    p = glm(LOW ~ AGE + LWT + RACE + SMOKE, family = binomial, data = train)
    np = randomForest( as.factor(LOW) ~ AGE + LWT + RACE + SMOKE, data = train,
                       replace = TRUE, mtry =4)
    risultato[i] = confronto_bg(test, p, np)
  }
  percentuale = mean(risultato)*100
  return (percentuale)
}
metodo_bg(dati, 0.5)
metodo_bg(dati, 0.90)
```

```
confronto_rf_bg= function(x, rf, bg){
  pred_rf = predict(rf, newdata = x, type = "prob")
  pred_rf1 = ifelse(pred_rf[,2] ==0, 0.001, pred_rf[,2])
  CV_rf =t(x$LOW) %*% log(pred_rf1) + t(1-x$LOW) %*% log(1-pred_rf1)
  pred_bg = predict(bg, newdata = x, type = "prob")
  pred_bg1 = ifelse(pred_bg[,2] ==0, 0.001, pred_bg[,2])
  CV_bg =t(x$LOW) %*% log(pred_bg1) + t(1-x$LOW) %*% log(1-pred_bg1)
  if ( CV_rf>=CV_bg)
  {return (1)}
  else
  {return (0)}
}

metodo_rf_bg = function(data, alfa) {
  n = 300
  risultato = rep(0,n)
  for (i in 1:n) {
    idx = sample(1:nrow(data), alfa*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    np1 = randomForest( as.factor(LOW) ~ AGE + LWT + RACE + SMOKE, data = train,
                        replace = FALSE)
    np2 = randomForest( as.factor(LOW) ~ AGE + LWT + RACE + SMOKE, data = train,
                        replace = TRUE, mtry = 4)
    risultato[i] = confronto_rf_bg(test, np1, np2)
  }
  percentuale = mean(risultato)*100
  return (percentuale)
}
metodo_rf_bg(dati, 0.5)
metodo_rf_bg(dati, 0.90)

##############################################
# Codice relativo ai dati di ‘‘Horseshoe Crab‘‘
# i dati sono reperibili usando il link https://users.stat.ufl.edu/~aa/cda/data.html
```

```
dati = read.table("Horseshoe Crab CDA agresti.txt", header = TRUE)
attach(dati)
crab.glm = glm(y ~  width , family = binomial)
summary(crab.glm)

# curva ROC
plot(roc(y, fitted(crab.glm)), print.auc = TRUE)

# test di Hosmer--Lemeshow
DescTools::HosmerLemeshowTest(fitted(crab.glm), y)

# test di Stukel
LogisticDx::stukel(crab.glm)

# Royston
royston1 = function(y, mod){
  pihat = fitted(mod)
  n = length(pihat)
  m = c(rep(0, n))
  pihat_s = sort(pihat, decreasing = TRUE)
  y_s = as.numeric(names(pihat_s))
  for (i in 1:n) {
    q = c(rep(0,i))
    for (j in 1:i) {
      q[j] = (y[y_s[j]] - pihat_s[j])
    }
    somma = - sum(q)
    m[i] = abs(somma)
  }
  return (max(m))
}
PR1 = royston1(y, crab.glm)
n = length(y)
PR1n = log(PR1)-0.5236*log(n)
m = sum(y)
L = log(m/(n-m))
```

```
mu1 = -1.0605-0.12684*(L^2)-0.010437*(L^3)
sigma1 = 0.3130+0.004261*L
1-pnorm(q = (PR1n-mu1)/sigma1)

royston2 = function(y, mod){
  pihat = fitted(mod)
  n = length(pihat)
  m = c(rep(0, n))
  pihat_s = sort(pihat, decreasing = TRUE)
  y_s = as.numeric(names(pihat_s))
  for (i in 1:(n)) {
    q = c(rep(0,i))
    for (j in 1:i) {
      q[j] = (y[y_s[j]] - pihat_s[j])
    }
    m[i] = -sum(q)
  }
  differenza = c(rep(0,n/2))
  for (i in 1:(n/2)){
    differenza[i] =abs(m[i]-m[n-i])
  }
  return (max(differenza))
}
PR2 = royston2(y, crab.glm)
PR2n = log(PR2)-0.5236*log(n)
mu2 = -1.1196-0.12093*(L^2)-0.009744*(L^3)
sigma2 = 0.3269+0.00654*L
1-pnorm(q = (PR2n-mu2)/sigma2)


# test di le Cessie
X = model.matrix(crab.glm)[,-1]
n = length(X)
p = 1
pihat = fitted(crab.glm)
e = y - pihat
```

```
V = diag(pihat * (1 - pihat))
V_m05 = diag( (pihat * (1 - pihat))^(-0.5) )
r_s = V_m05 %*% e
c_u = 0.5 * (4 / (n^(1/(2*p))))
s = sqrt(var(X))
# peso w_ij
w = function(x, i, j) {
  U = 0
  u = abs(x[i] - x[j]) / s
  U[u<=c_u] = 1
  return (U)
}
W = matrix ( 0, ncol = n, nrow = n)
for (i in 1:n) {
  for (j in 1:n) {
    W[i,j] = w(X, i, j)
  }
}
r_tilde = W %*% r_s
D_r = diag( diag(W %*% t(W)) )
T = (1/n)*t(r_tilde) %*% solve(D_r) %*% r_tilde
valori = media_emp(400,1000)   # valori usati anche nel dataset Low Birth Weight
mT = valori[1]    # stima della media
varT = valori[2]  # stima della varianza
b = 2*mT/varT
v = 2*(mT^2)/varT
# pvalue
1 - pchisq(q = b*T/n, df = v)



# metodo DRYV
library(randomForest)
library(tidymodels)
set.seed(123)
```

```
confronto_rf = function(x, mod, rf){
  pred_p = predict(mod, newdata = x ,type = "response")
  CV_p = t(x$y) %*% log(pred_p) + t(1-x$y) %*% log(1-pred_p)
  pred_np = predict(rf, newdata = x, type = "prob")
  pred_np1 = ifelse(pred_np[,2] %in% c(0, 1), c(0.001,0.999),pred_np[,2])
  CV_np =t(x$y) %*% log(pred_np1) + t(1-x$y) %*% log(1-pred_np1)
  if ( CV_p>=CV_np)
  {return (1)}
  else
  {return (0)}
}


metodo_rf = function(data, alfa) {
  n = 300
  risultato = rep(0,n)
  for (i in 1:n) {
    dx = sample(1:nrow(dati), alfa*nrow(dati))
    train = dati %>% slice(idx)
    test = dati %>% slice(-idx)
    if ( length(unique(train[,1]))==1 ) {
      next}
    p = glm(y ~ width, family = binomial, data = train)
    np = randomForest( as.factor(y) ~ width, data = train, replace = FALSE)
    risultato[i] = confronto_rf(test, p, np)
  }
  percentuale = mean(risultato*100
  return (percentuale)
}
metodo_rf(dati, 0.5)
metodo_rf(dati, 0.9)


confronto_bg = function(x, mod, bg){
  pred_p = predict(mod,newdata = x ,type = "response")
  CV_p = t(x$y) %*% log(pred_p) + t(1-x$y) %*% log(1-pred_p)
```

```
  pred_np = predict(bg, newdata = x, type = "prob")
  pred_np1 = ifelse(pred_np[,2] %in% c(0, 1), c(0.001,0.999),pred_np[,2])
  CV_np =t(x$y) %*% log(pred_np1) + t(1-x$y) %*% log(1-pred_np1)
  if ( CV_p>=CV_np)
  {return (1)}
  else
  {return (0)}
}

metodo_bg = function(data, alfa) {
  n = 300
  risultato = rep(0,n)
  for (i in 1:n) {
    idx = sample(1:nrow(data), alfa*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    if ( length(unique(train[,1]))==1 ) {
      next}
    p = glm(y ~ width, family = binomial, data = train)
    np = randomForest( as.factor(y) ~ width, data = train, replace = TRUE,
                       mtry =1)
    risultato[i] = confronto_bg(test, p, np)
  }
  percentuale = mean(risultato)*100
  return (percentuale)
}
metodo_bg(dati, 0.5)
metodo_bg(dati, 0.90)

confronto_rf_bg= function(x, rf, bg){
  pred_rf = predict(rf, newdata = x, type = "prob")
  pred_rf1 = ifelse(pred_rf[,2] %in% c(0, 1), c(0.001,0.999),pred_rf[,2])
  CV_rf = t(x$y) %*% log(pred_rf1) + t(1-x$y) %*% log(1-pred_rf1)
  pred_bg = predict(bg, newdata = x, type = "prob")
  pred_bg1 = ifelse(pred_bg[,2] %in% c(0, 1), c(0.001,0.999),pred_bg[,2])
  CV_bg =t(x$y) %*% log(pred_bg1) + t(1-x$y) %*% log(1-pred_bg1)
```

```
  # confronto
  if ( CV_rf>=CV_bg)
  {return (1)}
  else
  {return (0)}
}

metodo_rf_bg = function(data, alfa) {
  n = 300
  risultato = rep(0,n)
  for (i in 1:n) {
    idx = sample(1:nrow(data), alfa*nrow(data))
    train = data %>% slice(idx)
    test = data %>% slice(-idx)
    if ( length(unique(train[,1]))==1 ) {
      next}
    np1 = randomForest( as.factor(y) ~ width, data = train, replace = FALSE)
    np2 = randomForest( as.factor(y) ~ width, data = train,replace = TRUE, mtry = 1)
    risultato[i] = confronto_rf_bg(test, np1, np2)
  }
  percentuale = mean(risultato)*100
  return (percentuale)
}
metodo_rf_bg(dati, 0.5)
metodo_rf_bg(dati, 0.90)
```

Bibliografia

Agresti, A. (2013). *Categorical Data Analysis, 3rd ed.* John Wiley & Sons.

Agresti, A. (2015). *Foundations of Linear and Generalized Linear Models.* Wiley.

Breiman, L. & Cutler, A. (2023). *randomForest: Breiman and Cutler's Random Forests for Classification and Regression.* R package version 4.7-1.1.

Dardis, C. (2021). *LogisticDx: Diagnostic Tests for Models with a Binomial Response.* R package version 0.3.

Gasser, T. & Müller, H.-G. (1979). Kernel estimation of regression functions. In *Smoothing Techniques for Curve Estimation: Proceedings of a Workshop Held in Heidelberg, April 2–4, 1979.* Springer.

Hastie, T., Tibshirani, R., Friedman, J. H. & Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, vol. 2. Springer.

Hosmer, D. W., Hosmer, T., Le Cessie, S. & Lemeshow, S. (1997). A comparison of goodness-of-fit tests for the logistic regression model. *Statistics in Medicine* **16**, 965–980.

Hosmer, D. W. & Lemeshow, S. (1980). Goodness of fit tests for the multiple logistic regression model. *Communications in Statistics-Theory and Methods* **9**, 1043–1069.

Hosmer, D. W., Lemeshow, S. & Klar, J. (1988). Goodness-of-fit testing for the logistic regression model when the estimated probabilities are small. *Biometrical Journal* **30**, 911–924.

Hosmer, D. W., Lemeshow, S. & Sturdivant, R. X. (2013). *Applied Logistic Regression.* John Wiley & Sons.

Janková, J., Shah, R. D., Bühlmann, P. & Samworth, R. J. (2020). Goodness-of-fit testing in high dimensional generalized linear models. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **82**, 773–795.

le Cessie, S. & Van Houwelingen, H. C. (1995). Testing the fit of a regression model via score tests in random effects models. *Biometrics* **51**, 600–614.

le Cessie, S. & Van Houwelingen, J. (1991). A goodness-of-fit test for binary regression models, based on smoothing methods. *Biometrics* **47**, 1267–1282.

Lu, C. & Yang, Y. (2018). On assessing binary regression models based on ungrouped data. *Biometrics* **75**, 5–12.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications* **9**, 141–142.

Nattino, G., Pennell, M. L. & Lemeshow, S. (2020). Assessing the goodness of fit of logistic regression models in large samples: A modification of the Hosmer – Lemeshow test. *Biometrics* **76**, 549–560.

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C. & Müller, M. (2023). *pROC: Display and Analyze ROC Curves*. R package version 1.18.2.

Royston, P. (1992). The use of cusums and other techniques in modelling continuous covariates in logistic regression. *Statistics in Medicine* **11**, 1115–1129.

Royston, P. et al. (1993). Cusum plots and tests for binary variables. *Stata Technical Bulletin* **2**.

Salvan, A., Sartori, N. & Pace, L. (2020). *Modelli Lineari Generalizzati*. Springer.

Signorell, A. et al. (2023). *DescTools: Tools for Descriptive Statistics*. R package version 0.99.49.

Stukel, T. A. (1988). Generalized logistic models. *Journal of the American Statistical Association* **83**, 426–431.

Venables, W. N. & Ripley, B. D. (2023). *MASS: Support Functions and Datasets for Venables and Ripley's MASS*. R package version 7.3-60.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A* **26**, 359–372.

Wickham, H. et al. (2023). *tidyverse: Easily Install and Load the Tidyverse*. R package version 2.0.0.

Zhang, Y. & Yang, Y. (2015). Cross-validation for selecting a model selection procedure. *Journal of Econometrics* **187**, 95–112.