Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Magistrale in

Scienze Statistiche

Tesi di Laurea

Campagna pubblicitaria adattiva: un approccio bayesiano basato su sottogruppi

Relatore: Prof. Bruno Scarpa
Dipartimento di Scienze Statistiche

Laureanda: Valeria Zuccoli
Matricola: 1148583

Anno Accademico 2017/2018

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei simboli

- X_k, $k \in \{1, 2, ..., K\}$, generico biomarcatore
- $t \in \Omega = \{1, 2, ..., T\}$, trattamenti oggetto di studio
- $y \in \{0, 1\}$, variabile risposta
- $\boldsymbol{X}^{(n)} = \{\boldsymbol{x}_i\}_{i=1}^n$, caratteristiche molecolari dei primi n pazienti
- $\boldsymbol{z}^{(n)} = (z_1, ..., z_n)'$, trattamenti sottoposti ai primi n pazienti
- $\boldsymbol{y}^{(n)} = (y_1, ..., y_n)'$, risposte rilevate sui primi n pazienti
- $\mathcal{D}_n = \{\boldsymbol{X}^{(n)}, \boldsymbol{z}^{(n)}, \boldsymbol{y}^{(n)}\}$, informazioni disponibili dopo l'inserimento di n pazienti nel disegno
- $\Pi_r = \{S_1, S_2, ..., S_{M_r}\}$, generica partizione dello spazio dei biomarcatori
- $\boldsymbol{\Pi} = \{\Pi_1, ...\Pi_R\}$, spazio delle partizioni
- $n_m = \sum_{i=1}^n I(\boldsymbol{x}_i \in S_m)$, numero totale di pazienti nel sottogruppo S_m
- $n_{mt} = \sum_{i=1}^n I(\boldsymbol{x}_i \in S_m, z_i = t)$, numero totale di pazienti nel sottogruppo S_m assegnati al trattamento t
- $n_{mty} = \sum_{i=1}^n I(\boldsymbol{x}_i \in S_m, z_i = t, y_i = y)$, numero totale di pazienti nel sottogruppo S_m assegnati al trattamento t con risposta y
- n_{init}, ampiezza del campione iniziale
- ω, ampiezza della *wave*
- N, ampiezza del campione al termine del disegno
- N_{pop}, dimensione del dataset storico a disposizione

Introduzione

Il progresso della ricerca medica consente, già da diversi anni, di analizzare con sempre maggiore dettaglio gli elementi molecolari causa delle patologie umane. In oncologia, in particolare, tumori con la medesima collocazione e lo stesso aspetto macroscopico si rivelano spesso molto diversi per microstruttura e comportamento, evidenziando la necessità di terapie specifiche per particolari mutazioni genetiche o per marcatori prodotti. Considerato l'enorme numero di alterazioni possibili del DNA, però, non è semplice isolare manualmente un insieme di caratteristiche biologiche da impiegare come target per una singola terapia: si rende necessaria un'analisi automatica.

Negli ultimi anni la ricerca biostatistica si è quindi concentrata sulla costruzione di disegni sperimentali specifici per l'oncologia, tali da consentire il confronto tra due o più trattamenti su tumori macroscopicamente identici, ma con marcatori differenti. L'obiettivo di questi nuovi metodi è permettere un'analisi delle caratteristiche che meglio rispondono ad ogni cura sia a fini descrittivi (punto di partenza per studi biologici successivi) che inferenziali (previsione della miglior terapia per futuri pazienti).

Il presente elaborato si basa sul disegno adattivo basato sui sottogruppi (*Subgroup-Based Adaptive Design* - SUBA) proposto da Xu et al. (2016). La procedura ha inizio con una fase di assegnazione casuale delle cure ad un piccolo campione di pazienti. In seguito, viene selezionato un primo gruppo di individui che viene assegnato in maniera ragionata alla miglior terapia disponibile secondo le informazioni emerse in precedenza, proseguendo ad ondate successive fino ad esaurimento dei soggetti sperimentali. Questo consente, da un punto di vista etico, di migliorare a più persone possibili il decor-

so della malattia e, da un punto di vista economico-scientifico, di investire correttamente i fondi nelle terapie più adeguate.

L'impostazione adattiva di questo disegno è molto versatile e ne consente l'applicazione anche in settori differenti. In campo aziendale, ad esempio, esso può essere impiegato per lo sviluppo di campagne pubblicitarie personalizzate per ogni cliente tramite la definizione del miglior canale di contatto con cui promuovere un prodotto.

Nel caso in cui non siano disponibili dati su precedenti periodi promozionali, il metodo può seguire il campionamento in linea proposto in medicina. Il rapido apprendimento del modello consente un ridotto investimento in campagne "prova" effettuate su clienti casualmente scelti, seguito da un'allocazione intelligente di risorse nella fase ad ondate successive. Ciò si rivela tanto più utile quanto più sono volatili le preferenze dei consumatori rispetto al prodotto proposto; nel periodo economico presente, dominato dall'influenza immediata dei social network, il disegno riesce a tenere conto del rapido modificarsi dei gusti e delle opinioni.

Il metodo proposto, tuttavia, esprime le sue potenzialità anche per analizzare tramite simulazione dati già raccolti nel passato. Nel settore aziendale si ha spesso a disposizione una enorme massa di dati, che è una risorsa utilissima per la programmazione di strategie future. Come nel disegno sperimentale, la procedura è a più ondate successive, ma con la differenza che il "trattamento", ossia il canale di contatto, è già stato assegnato in precedenza e non è possibile intervenire per modificarlo. A tal proposito si è reso necessario un adattamento del metodo che consenta la gestione di osservazioni non manipolabili, sfruttando l'enorme dimensione della base di dati.

L'applicazione del disegno porta alla selezione ragionata di un sottoinsieme di osservazioni massimamente rappresentative degli individui che aderiscono alla campagna pubblicitaria. In sostanza, il disegno proposto si rivela essere un particolare metodo di campionamento per segmentare opportunamente la clientela di un'azienda. A differenza di altri metodi, esso sfrutta tutte le caratteristiche dei soggetti per isolare coloro che meglio definiscono il comportamento dei consumatori nel loro complesso. Ciò fornisce ai vertici

dell'azienda una descrizione dei propri consumatori ed inoltre rende disponibile una regola per l'assegnazione ottimale dei canali di contatto, utile per l'impostazione di una nuova campagna pubblicitaria.

L'elaborato è così strutturato: nel Capitolo 1 si descrive il disegno adattivo basato sui sottogruppi (*Subgroup-Based Adaptive Design* - SUBA) così come delineato in biostatistica da Xu et al. (2016); nel Capitolo 2 si propone un adattamento del metodo a dati storici di campagne di marketing aziendale; nel Capitolo 3, infine, si discute un caso reale di promozione bancaria su cui è stato applicato la metodologia proposta.

Capitolo 1

Disegno adattivo basato sui sottogruppi

In questo capitolo si sviluppa il disegno adattivo basato sui sottogruppi (*Subgroup-Based Adaptive design* – SUBA) come proposto da Xu et al. (2016). Questa metodologia, proposta dagli autori in ambito biostatistico, costituisce la base fondamentale per costruire una nuova procedura da applicare in campo pubblicitario.

1.1 Esperimenti clinici oncologici

Un esperimento clinico a coorte trasversale per il confronto tra terapie è generalmente costruito ripartendo casualmente un campione di pazienti in gruppi a cui vengono assegnate cure differenti. Questi individui vengono seguiti per un periodo prefissato al termine del quale si rileva la risposta dell'organismo (guarigione / non guarigione). L'attribuzione del trattamento al singolo soggetto avviene in maniera indipendente dagli altri pazienti, poiché la suddivisione nei gruppi avviene all'inizio della sperimentazione.

Sebbene questa procedura si riveli adeguata in molteplici ambiti, in oncologia la ricerca ha evidenziato che patologie apparentemente simili per collocazione e struttura macroscopica possono essere molto diverse dal punto di vista molecolare. Il medesimo tumore, infatti, può essere causato da svariate mutazioni genetiche che producono differenti biomarcatori rilevabili tramite

apposite analisi. Appare dunque necessario costruire sperimentazioni cliniche che si basino sulle peculiarità molecolari del tessuto studiato, costruendo sottogruppi di pazienti con caratteristiche simili cui sottoporre tutte le varie cure per ottenere un equo confronto. Terapie meno efficaci per un campione, tuttavia, possono rivelarsi ideali per un altro campione, in virtù delle differenze microscopiche tra gli individui. Obiettivo di un esperimento basato sui biomarcatori è isolare i gruppi di pazienti che meglio reagiscono ad ognuna delle cure esaminate, eventualmente evidenziando trattamenti costantemente meno efficaci di altri.

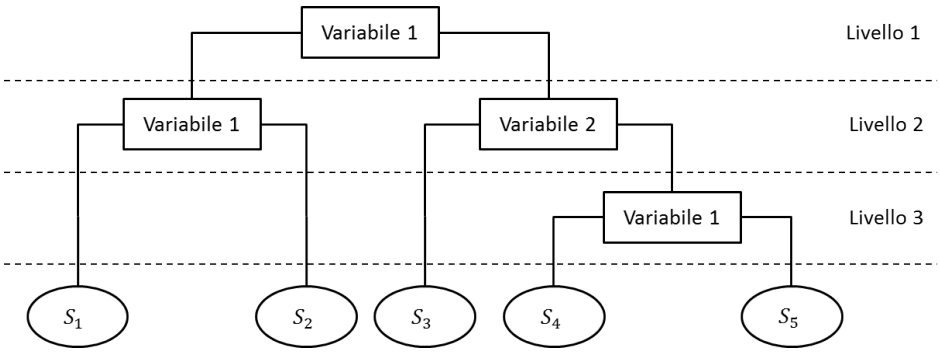
Considerata la dimensione ragguardevole del genoma umano, tuttavia, è facile comprendere come le combinazioni possibili di biomarcatori prodotti da un tessuto maligno siano molto numerose, per cui appare pressoché impossibile una classificazione manuale dei tumori. Si desidera perciò un metodo che consenta di distinguere le caratteristiche molecolari dei tessuti in gruppi omogenei per comportamento e, di conseguenza, per risposta alle terapie studiate.

Il disegno adattivo basato sui sottogruppi trattato in questo Capitolo rappresenta una possibile strada. Esso infatti restituisce, al termine della sperimentazione, sia una regola per l'assegnazione di nuovi pazienti al trattamento migliore, sia la descrizione della partizione ottimale dei biomarcatori.

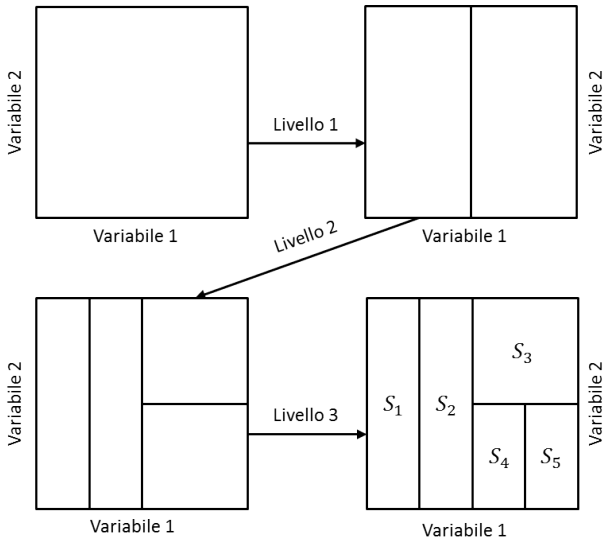
1.2 Partizioni

Punto di partenza è lo spazio dei biomarcatori, che, ipotizzando di aver rilevato K caratteristiche continue, coincide con $\mathbb{R}^K$. Come si è detto, l'intenzione è quella di costruire dei sottogruppi al cui interno si trovano individui con reazione omogenea ad una terapia prefissata. Evidentemente si desidera che questi insiemi non abbiano elementi in comune, poiché è necessario che un paziente sia collocato univocamente in uno solo di essi.

In matematica questo concetto ha il nome di *partizione*. Jech (2003) definisce partizione Π di uno spazio S una qualsiasi collezione di M sottoinsiemi disgiunti $\Pi = \{S_1, S_2, ..., S_M\}$ tali che la loro unione coincida con S stesso.



(a) Albero degli split



(b) Partizione del piano

Figura 1.1: Esempio di partizione

La costruzione di una partizione può essere realizzata in varie modalità. Nel metodo SUBA si propone uno schema ad albero non supervisionato, che garantisce la generazione di sottoinsiemi disgiunti e l'interpretabilità degli stessi. Dal nodo radice si procede secondo più livelli di divisione successivi: al livello iniziale si seleziona una variabile X_k con probabilità ν_k e la si impiega per una bipartizione dello spazio in base ad un punto arbitrario $\tilde{x}_k$, oppure è possibile rinunciare allo split con probabilità ν_0. Nel primo caso, al livello successivo si lavorerà sui sottoinsiemi $\{i : x_{ik} < \tilde{x}_k\}$ e $\{i : x_{ik} \geq \tilde{x}_k\}$; nel secondo caso, il nodo radice viene trasmesso intero al passo successivo.

Logicamente, i valori $\{\nu_k\}_{k=0}^{K}$ devono costituire una distribuzione propria di probabilità, per cui sono tali che, tra l'altro, $\sum_{k=0}^{K} \nu_k = 1$.

Nei livelli seguenti ogni nodo presente viene sottoposto alla medesima procedura della radice, generando alberi di profondità differente a seconda delle scelte effettuate.

Un esempio con due variabili e tre passi di divisione è rappresentato in Figura 1.1. Al primo livello è stata estratta per lo split la variabile X_1, generando due rami. Al secondo livello il ramo sinistro è stato diviso nuovamente per X_1, mentre il destro è stato suddiviso in base a X_2. Nell'ultimo passaggio, solo uno dei quattro insiemi è stato sottoposto a ulteriore split, secondo la variabile X_1.

Il punto di divisione $\tilde{x}_k$ può essere scelto in vari modi. Xu et al. (2016) propongono di utilizzare $\tilde{x}_k = med_k(P)$, dove $med_k(P)$ è la mediana della variabile X_k condizionata all'insieme P delle osservazioni disponibili al momento della divisione. Si noti che $med_k(P)$ rappresenta perciò un riferimento adattivo, destinato a modificarsi non appena nuovi dati saranno inseriti nella procedura.

Per quanto riguarda la profondità dell'albero della partizione, infine, gli autori suggeriscono di non superare i tre livelli di split, che coincidono con un numero massimo di otto sottogruppi. Questo sia per consentire interpretabilità delle partizioni che per evitare la costruzione di sottoinsiemi con un numero troppo piccolo di pazienti.

Costruite tutte le R possibili partizioni dello spazio dei biomarcatori, è possibile definire lo spazio delle partizioni $\mathbf{\Pi} = \{\Pi_1, ...\Pi_R\}$ come l'insieme che contiene ciascuna di esse.

Gli aspetti probabilistici legati alla partizione nel suo complesso saranno discussi nel paragrafo 1.5.2.

1.3 Tipologie di biomarcatori

Il disegno adattivo basato sui sottogruppi è stato inizialmente studiato per l'impiego di biomarcatori continui. Costruire partizioni in uno spazio $\mathbb{R}^K$

è infatti immediato.

L'estensione a variabili binarie è realizzabile con poco sforzo: è sufficiente imporre che la stessa caratteristica X_k non sia protagonista di divisioni successive. Una volta distinte le osservazioni con $X_k = 1$ da quelle con $X_k = 0$, infatti, non è più possibile effettuare ulteriori split basandosi su X_k, perché tutti i livelli della variabile sono stati separati.

Si sottolinea che in questi casi la scelta della variabile per la divisione del sottogruppo definisce univocamente lo split stesso. È scontato che proseguiranno nel ramo sinistro le osservazioni minori della unica soglia definita, mentre il lato destro accoglierà le unità maggiori o uguali.

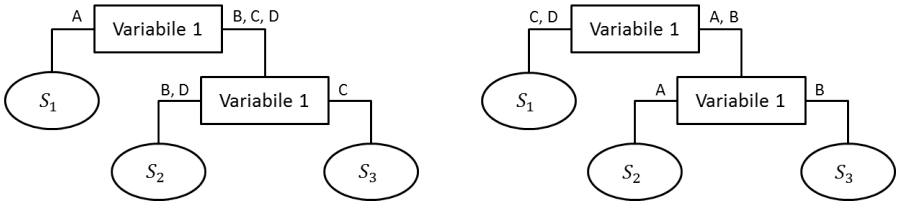


Figura 1.2: Esempi di albero con una sola variabile politomica

L'inserimento di variabili categoriali a più livelli risulta invece molto più complesso. Esse non sono sufficienti a definire univocamente la divisione, poiché, ignorando eventuali ordinamenti delle L modalità, esistono $L!(L-1)$ modi diversi di effettuare lo stesso split.

Nel caso di livelli ordinabili è possibile stabilire un criterio che emuli la scelta di $med_k(P)$ per le variabili continue, utilizzando la modalità mediana come punto di divisione.

Per caratteristiche sconnesse, invece, è necessario indicare sempre quali livelli proseguiranno nei due rami dell'albero, altrimenti non è possibile distinguere univocamente le partizioni. Si immagini, ad esempio, di voler effettuare una partizione sulla base di una sola caratteristica politomica a quattro modalità (A, B, C, D); se si trascurasse l'indicazione sui livelli, i due alberi in Figura 1.2 sarebbero indistinguibili poiché hanno la medesima struttura, quando in verità generano due partizioni decisamente diverse.

Inoltre si segnala che non è possibile stabilire il numero massimo realizzabile di divisioni successive per la singola variabile, poiché esso è condizionato agli split effettuati. Il sottogruppo S_2 della Figura 1.2a, infatti, potrebbe essere soggetto ad un'ulteriore divisione, poiché le unità al suo interno presentano due modalità (B e D); sarebbe quindi possibile effettuare uno split di terzo livello con la medesima variabile. Dall'altro lato, il sottogruppo S_2 corrispondente in Figura 1.2b contiene una sola modalità dell'esplicativa, quindi l'albero ammette al massimo due split consecutivi.

Nulla vieta di ricodificare le variabili qualitative tramite una opportuna serie di dicotomiche, che come si è visto sono gestite in maniera semplice. Questo passaggio però impone l'inserimento di numerose variabili nel modello, con un aumento di complessità non indifferente.

Riassumendo, qualsiasi sia il metodo con cui esse vengono inserite, le variabili politomiche portano un deciso incremento del costo computazionale e mnemonico rispetto alle continue. La dimensione dello spazio $\mathbf{\Pi}$ aumenta maggiormente, dunque anche le elaborazioni dei dati risultano più impegnative.

1.4 Linee guida del metodo

Il disegno sperimentale intreccia strettamente piano di campionamento, assegnazione al trattamento e modellazione dei dati.

Si supponga di avere a disposizione una popolazione di N soggetti sperimentali di cui si è interessati a rilevare la completa remissione del tessuto maligno. La variabile risposta è perciò di tipo binario.

Nella prima fase si estraggono n_{init} pazienti: essi vengono assegnati casualmente ai vari trattamenti $t \in \Omega = \{1, 2, ..., T\}$ e seguiti nel decorso della malattia. Successivamente, si campiona una prima ondata (*wave*) di $\omega \in \{1, ..., N - n_{init}\}$ pazienti, ma questa volta l'attribuzione delle terapie è condizionata alle risposte ottenute dal gruppo già trattato. Ciò è possibile mediante l'impiego di opportune densità descritte nei dettagli nel paragrafo 1.5. Al termine del periodo di follow-up si annotano le risposte dei sog-

getti della *wave* (guarigione / non guarigione) e si uniscono i dati raccolti al campione precedentemente esaminato.

A questo punto si estrae una nuova ondata di ω individui e si assegnano i trattamenti in maniera dipendente da quanto rilevato sui pazienti precedenti, seguendo quanto svolto per la prima *wave*. Si ripete perciò la procedura della prima ondata fino al raggiungimento di una prefissata numerosità N, oppure al soddisfacimento di una regola di interruzione precedentemente stabilita.

A conclusione del disegno si ottiene una regola per la miglior allocazione dei pazienti futuri, oltre alla descrizione della partizione dello spazio dei biomarcatori che ottimizza le assegnazioni ai trattamenti. I dettagli tecnici di questi risultati saranno trattati nei paragrafi 1.6 e 1.8.

L'ossatura del metodo può essere inoltre arricchita da una procedura di eliminazione anticipata di una terapia, delineata nel paragrafo 1.7. I trattamenti studiati possono, infatti, essere numerosi e talvolta accade che uno di questi abbia prestazioni inferiori agli altri qualunque siano i pazienti a cui viene assegnato. In questi casi si impone la rimozione della cura uniformemente meno efficace, procedendo nella sperimentazione con un insieme ridotto di trattamenti somministrabili. Ciò garantisce il principio etico base degli esperimenti clinici, secondo il quale i soggetti possono essere sottoposti solo a cure considerate equivalenti o migliori delle altre terapie (World Medical Association (2013), art. 33).

1.5 Costruzione delle distribuzioni

Dal punto di vista statistico è possibile analizzare distintamente il campionamento e il modello di assegnazione dei trattamenti.

Il campionamento avviene in stadi successivi. Anzitutto è necessario ottenere un insieme di N soggetti sperimentali, tramite una strategia opportunamente definita dai ricercatori. In seguito, la selezione del campione iniziale e delle ondate successive avviene secondo un campionamento casuale semplice, poiché tutti i soggetti hanno la medesima probabilità di essere estratti nelle varie fasi dello studio.

Il modello di attribuzione dei trattamenti, invece, è di tipo bayesiano con gerarchia tra i parametri. Il procedimento, infatti, combina le informazioni sulle partizioni dello spazio dei biomarcatori sotto forma di distribuzioni a priori con le risposte ottenute dai pazienti precedenti inserite tramite la verosimiglianza.

1.5.1 Funzione di verosimiglianza

Il modello probabilistico alla base del disegno è di tipo bernoulliano, poiché il singolo individuo i può manifestare risposta positiva ($y_i = 1$) o negativa ($y_i = 0$) alle cure cui viene sottoposto. Certamente la probabilità di successo sarà differente a seconda della partizione Π_r considerata ed al trattamento subito ($z_i = t$), perciò il singolo parametro scalare $\theta_{t,m}$ è così definito:

$$\theta_{t,m} = p(y_i = 1 | z_i = t, \Pi_r, \boldsymbol{x}_i \in S_m) \tag{1.1}$$

dove con $\boldsymbol{x}_i = (x_{i1}, ..., x_{iK})$ si intende il vettore delle esplicative del soggetto i-esimo e con $S_m \in \Pi_r$ un elemento della partizione.

Di conseguenza, la funzione di verosimiglianza congiunta per il vettore di parametri $\boldsymbol{\theta} = (\theta_{1,1}, ..., \theta_{T,1}, \theta_{1,2}, ..., \theta_{T,M})$ sarà il prodotto di densità bernoulliane. In ogni sottogruppo S_m si considerino:

- ◊ $n_m = \sum_{i=1}^{n} I(\boldsymbol{x}_i \in S_m)$, numero totale di pazienti
- ◊ $n_{mt} = \sum_{i=1}^{n} I(\boldsymbol{x}_i \in S_m, z_i = t)$, numero totale di pazienti assegnati al trattamento t
- ◊ $n_{mty} = \sum_{i=1}^{n} I(\boldsymbol{x}_i \in S_m, z_i = t, y_i = y)$, numero totale di pazienti assegnati al trattamento t con risposta y

dove con $I(\cdot)$ si intende la funzione indicatrice. La funzione di verosimiglianza assume dunque la seguente forma:

$$p(\boldsymbol{y}^{(n)} | \boldsymbol{X}^{(n)}, \boldsymbol{z}^{(n)}, \boldsymbol{\theta}, \Pi_r) = \prod_m \prod_t \theta_{t,m}^{n_{mt1}} (1 - \theta_{t,m})^{n_{mt0}} \tag{1.2}$$

dove si notano:

- ◊ $\boldsymbol{y}^{(n)} = (y_1, ..., y_n)'$, risposte rilevate sui primi n pazienti

$\Diamond$ $\boldsymbol{X}^{(n)} = \{\boldsymbol{x}_i\}_{i=1}^n$, caratteristiche molecolari dei primi n pazienti

$\Diamond$ $\boldsymbol{z}^{(n)} = (z_1, ..., z_n)'$, trattamenti sottoposti ai primi n pazienti

1.5.2 Distribuzioni a priori

Punto di partenza sono le partizioni e la rispettiva distribuzione a priori. Come si è visto nel paragrafo 1.2, esse sono costruite mediante uno schema ad albero che seleziona una variabile per volta con una data probabilità.

La probabilità della singola partizione, dunque, può essere costruita come proporzionale al prodotto delle probabilità ν_k delle variabili coinvolte negli split. Per la partizione proposta in Figura 1.1, ad esempio, il calcolo è così composto:

$$p(\Pi_r) \propto (\nu_1) \cdot (\nu_1\nu_2) \cdot (\nu_1\nu_0^3) = \nu_1^3\nu_2\nu_0^3 \tag{1.3}$$

dove in ogni parentesi si trova la probabilità di un livello di split.

Tuttavia, questo modello rende potenzialmente equiprobabili partizioni ottenute con un numero differente di variabili, mentre potrebbe essere desiderabile prediligere quelle che ne utilizzano un sottoinsieme ridotto. Ciò consentirebbe una migliore interpretazione dei risultati, evidenziando le caratteristiche molecolari più importanti nel trattamento della patologia. Viene dunque inserito un termine di penalizzazione proporzionale a ϕ^u, con $\phi \in (0, 1]$ e u pari al numero di variabili diverse impiegate negli split, allo scopo di rendere più probabili partizioni in cui lo stesso biomarcatore è utilizzato più volte nelle divisioni. Per l'esempio in Figura 1.1 si ha:

$$p(\Pi_r) \propto \nu_1^3\nu_2\nu_0^3 \cdot \phi^2 \tag{1.4}$$

Si noti che, nel caso limite con $\phi = 1$, l'Equazione 1.4 coincide con l'Equazione 1.3.

Costruite le a priori sulle partizioni, si passa all'analisi della probabilità di guarigione. In una partizione, ogni sottoinsieme comprende individui che possono potenzialmente essere assegnati a ciascuno dei trattamenti studiati. Ogni sottoinsieme, perciò, è caratterizzato da un vettore di parametri $\boldsymbol{\theta}_m = (\theta_{1,m}, ..., \theta_{T,m})$ che rappresenta la probabilità di successo relativa alle

differenti terapie. Le singole componenti possono avere differenti forme distributive a priori, ma vengono generalmente assunte $\theta_{t,m}|\Pi \sim Beta(a,b)$, con $f(\theta_{t,m};a,b) \propto \theta_{t,m}^{a-1}(1-\theta_{t,m})^{b-1}$, per coniugio con la verosimiglianza descritta nel paragrafo 1.5.1.

Infine è necessario un breve accenno agli iperparametri. La distribuzione a priori per le partizioni dipende dal vettore $\boldsymbol{\nu} = (\nu_k)_{k=0}^{K}$, che, in mancanza di informazione specifica sulla rilevanza clinica dei biomarcatori, può essere composto da elementi identici $\nu_k = \nu_j = \frac{1}{K+1}$, $\forall\, k,j \in \{0,...,K\}$. Per la probabilità di guarigione, invece, l'iperparametro coincide con il vettore $d = (a,b)$. Una a priori non informativa si ottiene fissando $d = (1,1)$, imposizione che consente alla Beta(a,b) di degenerare in una Uniforme$(0,1)$.

1.5.3 Distribuzione congiunta a posteriori

A questo punto è possibile costruire un modello bayesiano con gerarchia sui parametri che unisca le informazioni a priori sullo spazio delle partizioni e sulle probabilità di successo con i dati rilevati sui pazienti. Si definisce dunque una distribuzione congiunta a posteriori a tre livelli:

$$p(\boldsymbol{y}^{(n)}, \boldsymbol{\theta}, \Pi_r | \boldsymbol{X}^{(n)}, \boldsymbol{z}^{(n)}) \propto p(\boldsymbol{y}^{(n)} | \boldsymbol{X}^{(n)}, \boldsymbol{z}^{(n)}, \boldsymbol{\theta}, \Pi_r) p(\boldsymbol{\theta} | \Pi_r) p(\Pi_r) \qquad (1.5)$$

dove il livello base è dato dalla a priori sulla partizione $p(\Pi_r)$, da cui dipende la a priori sul vettore dei parametri $p(\boldsymbol{\theta}|\Pi_r)$ di cui, come ultimo passo, si ottiene la verosimiglianza $p(\boldsymbol{y}^{(n)}|\boldsymbol{X}^{(n)}, \boldsymbol{z}^{(n)}, \boldsymbol{\theta}, \Pi_r)$.

Questa funzione consente di raggiungere tutti gli obiettivi del disegno: l'inferenza sulla a posteriori relativa alle partizioni consente un'accurata analisi dei sottogruppi, mentre lo studio del vettore di parametri $\boldsymbol{\theta}$ permette l'esame del trattamento ottimale.

1.6 Allocazione ottima di nuovi pazienti

In questo studio è fondamentale stabilire una regola che definisca l'assegnazione migliore della terapia ad un soggetto nuovo, di cui sono disponibili

solo le informazioni sulle caratteristiche molecolari $\boldsymbol{x}_{n+1}$. L'idea è quella di costruire una funzione di probabilità predittiva a posteriori $q(t, \boldsymbol{x}_{n+1})$ che sia massimizzata dal trattamento ottimale. Essa può essere definita come segue:

$$\begin{aligned} q(t, \boldsymbol{x}_{n+1}) &\equiv p(y_{n+1} = 1 | \boldsymbol{x}_{n+1}, z_{n+1} = t, \mathcal{D}_n) \\ &= \sum_{\Pi_r \in \boldsymbol{\Pi}} q(t, \boldsymbol{x}_{n+1} | \Pi_r) \end{aligned} \tag{1.6}$$

dove con $\mathcal{D}_n = \{\boldsymbol{X}^{(n)}, \boldsymbol{z}^{(n)}, \boldsymbol{y}^{(n)}\}$ si denotano le informazioni disponibili prima dell'inserimento del paziente $n + 1$. Come si può osservare, essa è scomponibile in elementi derivanti dalle singole partizioni dello spazio, che per la generica Π_r assumono la forma seguente:

$$q(t, \boldsymbol{x}_{n+1} | \Pi_r) = p(y_{n+1} = 1 | \boldsymbol{x}_{n+1}, z_{n+1} = t, \Pi_r, \mathcal{D}_n) p(\Pi_r | \mathcal{D}_n) \tag{1.7}$$

ossia il prodotto tra la distribuzione a posteriori della partizione Π_r esaminata e la probabilità di successo condizionata sia ai dati disponibili ($\mathcal{D}_n$) che alle caratteristiche del paziente ($\boldsymbol{x}_{n+1}$).

Al soggetto $n + 1$ si attribuisce dunque il trattamento $z^*_{n+1} \in \Omega$ tale che:

$$z^*_{n+1} = \arg\max_{t \in \Omega} q(t, \boldsymbol{x}_{n+1}) \tag{1.8}$$

È necessario ora descrivere le componenti della funzione $q(t, \boldsymbol{x}_{n+1} | \Pi_r)$.

1.6.1 Distribuzione a posteriori per la partizione

Si consideri la partizione $\Pi_r = \{S_1, ..., S_{M_r}\}$. La distribuzione a posteriori è data da:

$$\begin{aligned} p(\Pi_r | \mathcal{D}_n) &\propto p(\Pi_r) p(\mathcal{D}_n | \Pi_r) \\ &= p(\Pi_r) \prod_m \prod_t \left\{ \int \prod_{\boldsymbol{x}_i \in S_m} p(y_i | \boldsymbol{x}_i, z_i = t, \theta_{t,m}) \, \mathrm{d}p(\theta_{t,m}) \right\} \end{aligned} \tag{1.9}$$

dove con $p(\Pi_r)$ si considera la densità a priori calcolata come nell'Equazione 1.4. Nel caso specifico di distribuzioni Beta(a, b) indipendenti impiegate

come priori per $\theta_{t,m}$, la posteriori per la partizione si semplifica come:

$$\begin{aligned} p(\Pi_r|\mathcal{D}_n) &\propto p(\Pi_r) \prod_m \prod_t \left\{ \int \theta_{t,m}^{n_{mt1}} (1-\theta_{t,m})^{n_{mt0}} \mathrm{Be}(\theta_{t,m}; a, b)\, \mathrm{d}\theta_{t,m} \right\} \\ &= p(\Pi_r) \prod_m \prod_t \frac{B(a+n_{mt1}, b+n_{mt0})}{B(a,b)} \end{aligned} \tag{1.10}$$

dove si evidenziano la funzione beta $B(a,b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$ e la densità $\mathrm{Be}(x; a, b) \propto x^{a-1}(1-x)^{b-1}$. Le definizioni di n_m e n_{mty} sono le medesime del paragrafo 1.5.1.

1.6.2 Probabilità condizionata di successo

Considerando per i parametri $\theta_{t,m}$ una a priori $\mathrm{Beta}(a,b)$, la probabilità di successo condizionata ai dati disponibili $\mathcal{D}_n$ non è altro che l'integrale del parametro stesso rispetto alla sua distribuzione a posteriori $\mathrm{Beta}(a+n_{mt1}, b+n_{mt0})$:

$$\begin{aligned} p(y_{n+1} = 1|\boldsymbol{x}_{n+1}, z_{n+1} = t, \Pi_r, \mathcal{D}_n) &= \sum_m I(\boldsymbol{x}_{n+1} \in S_m) \int \theta_{t,m} \mathrm{d}p(\theta_{t,m}|\Pi_r, \mathcal{D}_n) \\ &= \sum_m I(\boldsymbol{x}_{n+1} \in S_m) \frac{a+n_{mt1}}{a+b+n_{mt}} \end{aligned} \tag{1.11}$$

Evidentemente, poiché il soggetto $n+1$ appartiene ad uno ed un solo sottoinsieme S_m, l'Equazione 1.11 si riduce ad un solo termine.

1.7 Eliminazione anticipata di trattamenti

Il metodo di allocazione dei pazienti appena descritto consente potenzialmente di costruire una graduatoria delle terapie per ogni soggetto sperimentale. Può succedere, soprattutto con un numero T di possibilità molto ampio, che una cura si dimostri sempre meno efficace delle altre e sia opportuno rimuoverla dalla sperimentazione. Ad ogni ondata di pazienti, dunque, si devono sottoporre tutti i trattamenti ad una verifica empirica, al fine di

verificare la loro ottimalità almeno in un sottospazio di $\mathbb{R}^K$. Questo procedimento non può essere effettuato sfruttando solamente i dati dei pazienti già esaminati, poiché è realistico che non coprano tutto lo spazio di variazione dei biomarcatori, ma serve uno studio analitico.

Si consideri la generica caratteristica molecolare k e i valori osservati $\{x_{1k}, ..., x_{nk}\}$ di cui $\min_k$ e $\max_k$ sono rispettivamente il minimo e il massimo. Si definisce una griglia equispaziata di dimensione H_0 sull'intervallo $[\min_k, \max_k]$ e si ripete per ogni biomarcatore $k \in \{1, ..., K\}$. Il prodotto cartesiano di queste sequenze crea una griglia K-dimensionale $\tilde{\boldsymbol{x}} = \{\tilde{x}_h \in \mathbb{R}, h = 1, ..., H\}$ di ampiezza $H = H_0^K$. I valori inseriti nella griglia fungono da pazienti fittizi, per i quali viene previsto il trattamento ottimale: qualora uno di essi si riveli uniformemente meno efficace degli altri, si provvederà alla sua rimozione dal disegno. In simboli, la terapia t^* tale per cui:

$$q(t^*, \tilde{x}_h) < q(t, \tilde{x}_h), \qquad \forall\, h = 1, ..., H, \;\; \forall\, t \neq t^* \tag{1.12}$$

viene eliminata ($\Omega \equiv \Omega \setminus \{t^*\}$). Logicamente, nel caso in cui resti un solo trattamento all'interno dell'insieme Ω, il disegno viene interrotto anticipatamente, con la descrizione della partizione nulla e l'assegnazione ottimale dei soggetti all'unica cura rimasta.

1.8 Descrizione della partizione ottimale

Come si è detto, il metodo può riportare, oltre alla regola di miglior allocazione dei pazienti, la descrizione della partizione più efficace nell'attribuzione dei trattamenti. Si desidera perciò ottenere una partizione che sintetizzi l'intera distribuzione $p(\Pi_r|\mathcal{D}_n)$ emersa dal disegno, come si farebbe nel caso di scalari calcolando un semplice indice di posizione.

Seguendo questa logica si può costruire un procedimento che estragga la partizione media dello spazio $\boldsymbol{\Pi}$ rispetto alla distribuzione a posteriori $p(\Pi_r|\mathcal{D}_n)$, tramite la costruzione di opportune matrici di associazione.

Ad ogni partizione Π_r, infatti, si fa corrispondere una matrice di dimensione $N \times N$ di associazione tra gli individui G^{Π_r}. L'elemento $G_{ij}^{\Pi_r}$ assume valore pari a 1 se le unità appartengono al medesimo sottoinsieme S_m della partizione Π_r, mentre è pari a 0 se sono parte di due sottoinsiemi diversi $S_m \neq S_n$. In simboli:

$$G_{ij}^{\Pi_r} = \begin{cases} 0, & \text{se } \boldsymbol{x}_i, \boldsymbol{x}_j \in S_m \\ 1, & \text{se } \boldsymbol{x}_i \in S_m \wedge \boldsymbol{x}_j \in S_n,\ m \neq n \end{cases} \tag{1.13}$$

con $S_m, S_n \in \Pi_r$. Si noti che la matrice è simmetrica rispetto alla diagonale principale poiché necessariamente $G_{ij}^{\Pi_r} = G_{ji}^{\Pi_r}$. Questo procedimento genera l'insieme di tutte le possibili matrici di associazione $G^{\mathbf{\Pi}} = \{G^{\Pi_1}, ..., G^{\Pi_R}\}$.

Si passa quindi al calcolo della matrice di associazione media rispetto alla distribuzione a posteriori $p(\Pi_r|\mathcal{D}_n)$:

$$\bar{G} = \sum_{r=1}^{R} G^{\Pi_r} p(\Pi_r|\mathcal{D}_n) \tag{1.14}$$

Nel caso di scalari, la media può non coincidere con alcuno tra i valori rilevati: ad esempio, se $\alpha = \{3, 4\}$ si avrà $\bar{\alpha} = 3.5$ che non corrisponde ad alcuna unità nel campione. Analogamente, la matrice $\bar{G}$ può essere diversa da ciascuna delle G^{Π_r} calcolate, dunque è necessario estrarre quella che risulti ad essa più simile. Xu et al. (2016) propongono una selezione basata sui minimi quadrati:

$$G^{LS} = \arg \min_{G^{\Pi_r} \in G^{\mathbf{\Pi}}} \|G^{\Pi_r} - \bar{G}\|^2 \tag{1.15}$$

dove con $\|A\|^2$ si intende la somma dei quadrati degli elementi della matrice A. La partizione corrispondente Π^{LS} minimizza la somma degli scarti quadratici tra la sua matrice di associazione G^{LS} e quella della media a posteriori $\bar{G}$ e può essere considerata partizione ottimale.

Naturalmente, la funzione obiettivo qui impiegata non è l'unica possibile. Xu et al. (2016) sottolineano che l'Equazione 1.15 trova giustificazione in virtù della regola di Bayes, ma è piuttosto impegnativa dal punto di vista computazionale. A questo scopo propongono una via più veloce, mini-

mizzando la media delle deviazioni quadratiche rispetto alla distribuzione a posteriori $p(\Pi_r|\mathcal{D}_n)$:

$$G_*^{LS} = \arg \min_{G^{\Pi_*} \in G^{\mathbf{\Pi}}} \sum_{r=1}^{R} \|G^{\Pi_r} - G^{\Pi_*}\|^2 p(\Pi_r|\mathcal{D}_n) \tag{1.16}$$

dove la notazione $\|\cdot\|^2$ è analoga a quanto indicato precedentemente.

Si noti, tuttavia, che funzioni obiettivo diverse possono estrarre partizioni differenti, dunque sarà compito del ricercatore scegliere opportunamente la procedura più adatta al caso specifico.

Capitolo 2

Adattamento del disegno a dati già raccolti

Come si è visto, al termine del disegno si ottengono due risultati: una regola per l'assegnazione ottimale al trattamento e la descrizione della partizione che meglio descrive le differenze tra i pazienti.

Sebbene la procedura sia nata in campo biostatistico, nulla impedisce di applicarla altrove, ad esempio in ambito aziendale. Si immagini quindi di voler segmentare la clientela di un'impresa, in modo da meglio descriverne le caratteristiche utili a pianificare una campagna pubblicitaria con diversi canali di contatto. I canali di contatto (o strategie) prendono il posto dei trattamenti utilizzati in medicina.

A tal proposito, si supponga di avere già disponibile un enorme dataset in cui sono state raccolte le informazioni relative ad una precedente promozione aziendale. In dati di questo genere la percentuale di individui che ha sottoscritto il prodotto proposto è sempre molto piccola. Ogni analisi richiede un accurato bilanciamento, ma solitamente questa operazione è effettuata in maniera casuale, causando l'eliminazione di molte osservazioni potenzialmente rilevanti. L'adattamento del disegno adattivo basato sui sottogruppi ai dati già raccolti consente, invece, la selezione di un campione ragionato entro cui sono presenti le unità che meglio consentono la descrizione della popolazione iniziale.

In questo Capitolo verrà presentata la metodologia per costruire con

questa logica un campione di ampiezza N partendo da N_{pop} osservazioni disponibili.

2.1 Simulazione del disegno

La fase iniziale di allenamento risulta piuttosto semplice da replicare: è sufficiente estrarre n_{init} soggetti, rilevare il canale di contatto e la variabile risposta e procedere con il calcolo della distribuzione a posteriori. È complesso, invece, proseguire nella fase ad ondate successive, perché in ogni *wave* si dovrebbe effettuare un intervento sui soggetti che la compongono, cioè si dovrebbe assegnare loro la miglior strategia. Se questo non fosse necessario, sarebbe sufficiente campionare gli individui, prendere nota della variabile di interesse e ripetere. Nei dati già raccolti il contatto è già stato effettuato, ma è possibile sfruttare le potenzialità del dataset originale, molto più grande dell'insieme che si intende costruire.

Per la creazione dell'ondata, dunque, si procede con un campionamento iniziale di ampiezza ω, così da selezionare i clienti da analizzare. In seguito, essi vengono sottoposti alla prescrizione del canale ottimale di contatto, che può essere diverso da quello realmente utilizzato. Le unità correttamente allocate vengono conservate, mentre le altre necessitano di essere sostituite.

2.2 Sostituzione delle unità

Per ogni osservazione $\boldsymbol{x}$ sottoposta al tipo di contatto sbagliato si cerca idealmente una sostituta $\boldsymbol{x}^*$ con caratteristiche identiche, con l'eccezione del canale utilizzato. Poiché una esatta corrispondenza risulta pressoché impossibile, è necessario stabilire un metodo che determini l'unità più simile a $\boldsymbol{x}$ che sia stata sottoposta alla strategia prevista t^*.

2.2.1 Elemento della partizione base

Una strada è quella di sostituire il soggetto con un altro presente nel medesimo elemento della partizione base. Essa è costituita dalle intersezioni

degli insiemi delle singole partizioni: ogni sottogruppo, infatti, è generato dall'unione di almeno due elementi base. Si sottolinea che la partizione base non è compresa nello spazio $\mathbf{\Pi}$ impiegato nel disegno, poiché non deriva dallo stesso processo ad albero delle altre bensì dall'intersezione di tutte le Π_r.

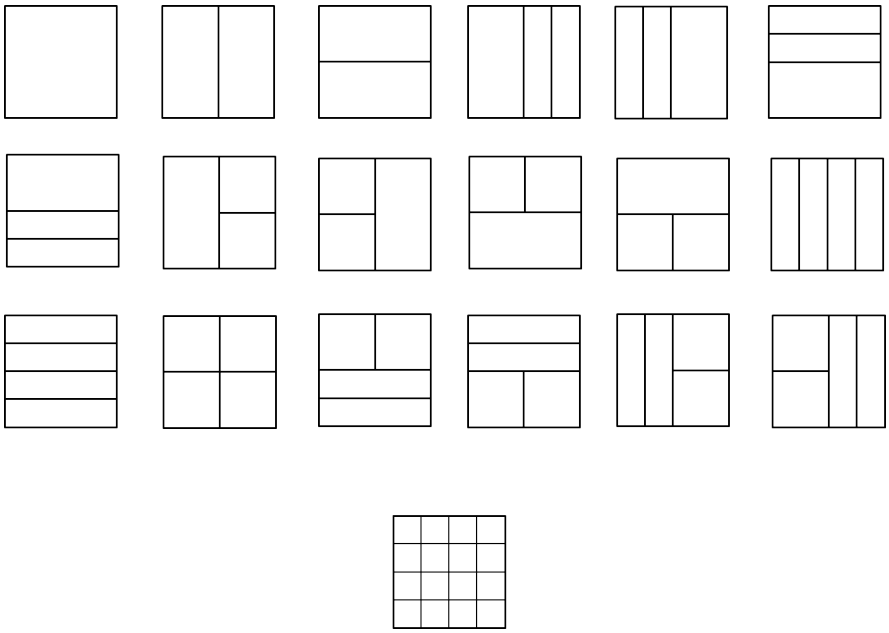


Figura 2.1: Partizioni di un piano generate da due livelli di split successivi

Nell'esempio in Figura 2.1 si riportano le 18 partizioni di un piano ottenute tramite solo due livelli di divisione. Come si può notare, la sovrapposizione di tutte le partizioni genera la partizione base ai piedi della figura. Ciò garantisce che ai fini del modello l'unità sostitutiva sia equivalente all'originale sotto ogni punto di vista, poiché si trova nello stesso sottoinsieme in tutte le possibili partizioni.

Questo metodo, tuttavia, si rivela irrealizzabile a causa della maledizione della dimensionalità: anche con un numero di variabili piuttosto ridotto, infatti, gli elementi della partizione base si rivelano moltissimi, per cui trovare un soggetto che condivida il medesimo spazio può rivelarsi impossibile.

2.2.2 Distanza tra le unità

Per approssimare il processo appena descritto, è possibile impiegare misure di similarità o distanza tra le unità. Si sceglie come sostituto l'individuo $\boldsymbol{x}^*$ che abbia la minima distanza da quello di interesse, caratterizzato dal trattamento corretto t^*:

$$\boldsymbol{x}^* = \arg \min_{\boldsymbol{x}^*:\, z^*=t^*} dist(\boldsymbol{x}, \boldsymbol{x}^*) \tag{2.1}$$

dove con $dist(\cdot, \cdot)$ si intende un'opportuna funzione di distanza. Nel caso di biomarcatori continui o binari si ritiene giustificato l'utilizzo della metrica euclidea, mentre l'inserimento di variabili politomiche renderebbe necessario l'utilizzo di una funzione di similarità opportuna.

Anche questo metodo, tuttavia, presenta un problema: talvolta per due soggetti diversi si seleziona come alternativa la medesima osservazione. Da un punto di vista logico è possibile imporre sostituti sempre diversi: ipotizzando che le unità $\boldsymbol{x}_1$ e $\boldsymbol{x}_2$ siano distanti dall'osservazione $\boldsymbol{x}^*$ rispettivamente $dist(\boldsymbol{x}_1, \boldsymbol{x}^*)$ e $dist(\boldsymbol{x}_2, \boldsymbol{x}^*) = dist(\boldsymbol{x}_1, \boldsymbol{x}^*)+\varepsilon$, si richiede che $\boldsymbol{x}_2$, il più lontano, cerchi un'alternativa.

In caso di ondate piuttosto numerose, però, può succedere che la stessa sostituta rappresenti anche più di due unità, comportando un incremento computazionale non indifferente. A questo proposito si suggerisce semplicemente di eliminare le osservazioni duplicate dalla lista delle alternative, mantenendone dunque una singola copia.

Ciò introduce un errore nel processo, poiché la numerosità della *wave* non corrisponde più a quella nominale; tuttavia, essendo la simulazione in corso un'approssimazione del disegno adattivo basato sui sottogruppi, si ritiene trascurabile l'entità di questa imprecisione.

Capitolo 3

Promozione di prodotti bancari

Per dimostrare la validità del disegno proposto nell'analisi di segmentazione lo si è impiegato in un caso reale di promozione di prodotti bancari. L'applicazione ha alla base un opportuno insieme di dati già raccolti, al fine di utilizzare l'estensione descritta nel Capitolo 2.

Per l'analisi è stato impiegato il software statistico R (R Core Team (2018)). In Appendice A.1 sono indicate le versioni del software e i pacchetti utilizzati.

3.1 Descrizione dei dati

È stato utilizzato il dataset *Bank Marketing* disponibile liberamente su UCI Machine Learning Repository (2012). Esso è relativo ad una campagna pubblicitaria condotta da un'istituzione bancaria portoghese per promuovere un deposito a termine su 41 188 soggetti.

La risposta è di tipo dicotomico, dove il valore 1 rappresenta la sottoscrizione del prodotto da parte del cliente e il valore 0 il rifiuto della proposta. La percentuale di adesioni alla campagna è pari al 11,26%.

È disponibile una variabile relativa al canale tramite il quale è stata effettuata la promozione (cellulare/telefono fisso): essa è stata scelta come fosse il "trattamento" a cui sono stati sottoposti i soggetti. Di conseguenza si avrà $t \in \Omega = \{1, 2\}$.

Le esplicative sono 19. Tra esse si distinguono informazioni relative al cliente specifico:

- Età (numerica)
- Professione (categoriale, 12 livelli)
- Stato civile (categoriale, 3 livelli: *sposato*, *divorziato/vedovo*, *celibe/-nubile*)
- Livello di istruzione (categoriale, 8 livelli)
- Credito in default (categoriale, 3 livelli: *no*, *sì*, *sconosciuto*)
- Presenza di mutuo (categoriale, 3 livelli: *no*, *sì*, *sconosciuto*)
- Presenza di prestito personale (categoriale, 3 livelli: *no*, *sì*, *sconosciuto*)

altre riguardanti il contatto per la presente campagna promozionale:

- Mese dell'ultimo contatto (categoriale, 12 livelli)
- Giorno della settimana dell'ultimo contatto (categoriale, 5 livelli)
- Durata dell'ultimo contatto in secondi (numerica)
- Numero di contatti in questa campagna promozionale (numerica)

o per campagne precedenti:

- Giorni passati dopo l'ultimo contatto per la campagna precedente (numerica, dove 999 significa nessun contatto precedente)
- Numero di contatti con il cliente precedenti la campagna in oggetto (numerica)
- Risultato della campagna precedente (3 livelli: *non esistente*, *rifiuto*, *successo*)

ed infine variabili attinenti il contesto socioeconomico al momento della proposta:

- Variazione del tasso di occupazione (numerica, indicatore trimestrale)
- Indice dei prezzi al consumo (numerica, indicatore mensile)
- Indice di fiducia dei consumatori (numerica, indicatore mensile)
- Tasso Euribor a tre mesi (numerica, indicatore giornaliero)

- Numero di posti di lavoro (numerica, indicatore mensile)

Obiettivo dell'applicazione presentata è simulare il metodo secondo l'estensione del Capitolo 2 per confrontarlo con il campionamento casuale semplice. Inoltre si studierà la sensibilità della procedura ai diversi valori dei parametri (numero di unità iniziali, ampiezza della singola ondata, dimensione del campione finale...).

3.2 Trasformazione e selezione di variabili

Anzitutto sono state eliminate le informazioni relative alla durata dell'ultima chiamata e al numero di contatti effettuati durante la campagna oggetto di studio: esse sono *leaker*, poiché strettamente correlate alla risposta e non disponibili in fase di pianificazione del progetto pubblicitario.

In seguito sono state tralasciate le esplicative politomiche non ricodificabili come binarie, cioè la professione e il livello di istruzione, il giorno della settimana e il mese di contatto. Questo perché, come analizzato nel paragrafo 1.3, la complessità del problema sarebbe stata eccessiva e la sua gestione esula dall'obiettivo principale di valutare la procedura proposta.

Per lo stato civile si è deciso di accorpare i livelli *sposato* e *divorziato-vedovo*, poiché rappresentano persone che hanno avuto esperienza di coppia strutturata; la variabile assume perciò i due livelli *mai sposato* e *altro*.

Le variabili relative alla presenza di un prestito personale e di un mutuo presentano valori *sconosciuto* per le medesime 987 osservazioni: si è deciso di eliminarle perché esse sono un numero esiguo e non si ritiene influenzino il risultato. Per lo stesso motivo sono state eliminate 3 osservazioni con presenza di credito in default, lasciando solamente i livelli *no* e *sconosciuto*.

Quanto alle informazioni sulla campagna precedente, si è scelto di mantenere una sola variabile che indicava un precedente contatto (*sì* o *no*), eliminando tutte le altre esplicative al riguardo: i soggetti già sottoposti ad una campagna di marketing sono solamente 1 515, dunque ulteriori livelli di dettaglio (numero di telefonate, risultato della promozione) sarebbero apparsi superflui.

Al termine di questo passaggio sono disponibili 11 esplicative e 40 198 osservazioni. La percentuale di adesioni alla campagna è pari al 11,28%, pressoché identica al dataset originale.

3.2.1 Selezione di variabili

Come si è visto, il disegno diventa estremamente complesso anche con un numero piuttosto irrisorio di esplicative. Essendo interessati esclusivamente alla valutazione della procedura, si è preferito procedere ad un'ulteriore riduzione della dimensionalità del problema a 8 variabili, per non incorrere in problemi computazionali.

Per questo motivo, il dataset è stato bilanciato e le esplicative rimaste sono state inserite in diversi modelli classici di classificazione al fine di estrapolare le più importanti.

Variabile	Modello logistico	An. discrim. lineare	MARS	Rete neurale	Foresta casuale
Età	0.4323	0.4246	0.1003	0.0000	0.2801
Stato civile	0.5938	0.5829	0.0000	0.1513	0.0309
Default	1.0000	1.0000	0.1429	0.4821	0.1628
Mutuo	0.0000	0.0041	0.0000	0.1195	0.0000
Prestito	0.0914	0.0806	0.0000	0.3482	0.0014
Prec. contatto	0.2019	0.1673	0.0000	0.3499	0.2896
Variaz. tasso occup.	0.7278	0.7732	0.3185	0.4239	0.6660
Indice prezzi	0.5524	0.5834	0.0676	0.2714	0.2956
Fiducia consumatori	0.5144	0.5311	0.3941	0.2322	0.4457
Euribor	0.1527	0.0000	1.0000	0.9810	0.7933
N. impiegati	0.7209	0.6335	0.4154	1.0000	1.0000

Tabella 3.1: Importanza delle variabili nei singoli modelli

Non essendo nota a priori la struttura dei dati, sono stati scelti modelli che classificassero le unità sulla base di strutture geometriche molto diverse. I parametri di regolazione sono stati ottimizzati tramite la divisione delle

osservazioni in insiemi di stima e verifica, al fine di limitare qualsiasi effetto di sovradattamento.

Per ogni modello è stata estratta l'importanza di ciascuna variabile secondo la proposta di Kuhn (2018), rappresentata dagli indicatori seguenti:

- Modello logistico: valore assoluto della statistica test t
- Analisi discriminante lineare: valore assoluto della statistica test t
- Multivariate Adaptive Regression Splines (MARS): numero di sottoinsiemi determinati da questa variabile
- Rete neurale: combinazioni opportune dei pesi degli archi in valore assoluto
- Foresta casuale: riduzione media dell'impurità del nodo secondo l'indice di Gini

e, dopo un passaggio di normalizzazione, si è ottenuto quanto riportato in Tabella 3.1.

Questi indicatori sono stati infine sintetizzati tramite una media aritmetica, così da ottenere una classifica per importanza delle variabili secondo la Tabella 3.2.

Variabile	Importanza media
N. impiegati	0.7540
Euribor	0.5854
Variaz. tasso occup.	0.5819
Default	0.5576
Fiducia consumatori	0.4235
Indice prezzi	0.3541
Stato civile	0.2718
Età	0.2474
Prec. contatto	0.2017
Prestito	0.1043
Mutuo	0.0247

Tabella 3.2: Importanza media delle variabili

Come si è detto, per motivi computazionali è necessario limitare il numero a $K = 8$ esplicative, per cui le informazioni su un eventuale contatto precedente e sulla presenza di prestito o mutuo sono state tralasciate nello sviluppo dell'analisi.

Si noti che questo processo di selezione è stato sviluppato in carenza di conoscenze specifiche riguardo all'azienda del caso. Sarebbe stato possibile applicare anche procedure diverse, tuttavia quella presentata consente la valutazione delle variabili combinando metodi che sfruttano strutture geometriche molto differenti. In ambito aziendale è sempre preferibile, laddove possibile, una selezione ragionata basata sull'esperienza, come indicato da Azzalini & Scarpa (2012).

3.2.2 Esplicative selezionate

Al termine del processo di selezione e trasformazione, vengono quindi scelte per l'analisi le seguenti variabili:

- Età
- Stato civile (2 livelli: *single*, *non single*)
- Credito in default (2 livelli: *no*, *sconosciuto*)
- Variazione del tasso di occupazione
- Indice dei prezzi al consumo
- Indice di fiducia dei consumatori
- Tasso Euribor a tre mesi
- Numero di posti di lavoro
- Trattamento: canale di contatto durante la campagna in oggetto (2 livelli: *cellulare*, *telefono fisso*)
- Risposta: risultato della campagna pubblicitaria (2 livelli: *sottoscrizione*, *non sottoscrizione*)

tutte numeriche, dove non indicato espressamente.

Infine, le esplicative sono state opportunamente normalizzate per limitare l'influenza dell'aspetto dimensionale nel calcolo delle distanze tra unità.

La normalizzazione sembra la via più adatta in virtù della presenza di variabili binarie, che assumono così il medesimo campo di variazione delle quantitative.

3.3 Simulazione del disegno

Come si è visto, la simulazione su dati storici prevede più passaggi:

- Costruzione dello spazio delle partizioni $\mathbf{\Pi}$ e dei sottogruppi
- Allenamento del modello sul campione iniziale di n_{init} soggetti
- Ripetizione, fino al raggiungimento di N soggetti sperimentali, di:
  - Selezione di un'ondata di ω clienti e prescrizione del canale ottimale
  - Sostituzione delle unità con assegnazione errata tramite il metodo delle distanze
  - Nuovo allenamento del modello

Al termine della sperimentazione si ottengono la distribuzione congiunta a posteriori e gli indici dei soggetti entrati a far parte del campione.

Nell'analisi qui riportata non è stata applicata la procedura di eliminazione anticipata dei trattamenti descritta nel paragrafo 1.7 poiché sono disponibili solamente due canali di contatto pubblicitario.

3.3.1 Costruzione delle partizioni

La costruzione delle partizioni avviene in modo indipendente dai valori dei parametri n_{init}, ω e N. L'inizializzazione del modello, infatti, richiede solamente tre argomenti:

- Numero K di esplicative considerate
- Tipologia di ognuna di esse (numerica o binaria)
- Numero T di livelli della variabile che funge da trattamento

Per prima cosa viene costruita la matrice delle partizioni. Ciascuna di esse è rappresentata da una riga di lunghezza pari a 7, numero di nodi dell'albero più profondo: ogni elemento è una variabile impiegata nella divisione corrispondente. In questo modo la partizione è definita univocamente, come si è discusso nel paragrafo 1.3.

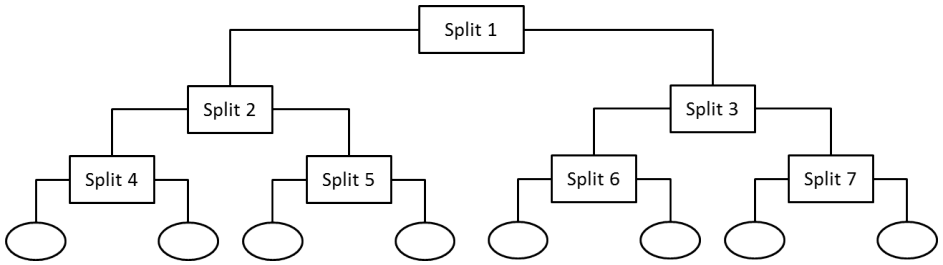


Figura 3.1: Numerazione dei nodi dell'albero

La corrispondenza tra la posizione del singolo split e l'elemento della matrice è riportata in Figura 3.1. È evidente che se in posizione 2 non viene effettuata alcuna divisione, dando così origine ad una foglia, necessariamente anche gli elementi di posto 4 e 5 devono essere vuoti; analogo discorso vale per le posizioni 1 e 3. Si simboleggia l'assenza di split con il valore 0. L'esempio in Figura 1.1 può dunque essere indicato con il vettore $(1, 1, 2, 0, 0, 0, 1)$.

Secondariamente viene creata la matrice dei sottogruppi: ogni partizione è infatti una collezione di sottoinsiemi disgiunti, definiti univocamente dalle variabili impiegate per le divisioni successive e dal ramo scelto. La singola riga è quindi composta da 6 elementi, dove nei posti dispari trovano collocazione le esplicative responsabili degli split nei tre livelli e nei posti pari l'indicazione della direzione secondo cui proseguire (destra/sinistra). Anche in questo caso, l'assenza di divisione è simboleggiata dal valore 0. I sottoinsiemi della partizione in Figura 1.1 potrebbero essere codificati come in Tabella 3.3.

In seguito, i sottogruppi vengono messi in corrispondenza con le partizioni tramite un'altra matrice, dove ogni riga contiene l'indicazione dei sottogruppi relativi ad una singola partizione. La matrice avrà perciò 8 colonne, pari al numero massimo di foglie dell'albero più profondo. Nel caso in cui gli

Sottogruppo	Livello 1	Ramo	Livello 2	Ramo	Livello 3	Ramo
S_1	1	sx	1	sx	0	0
S_2	1	sx	1	dx	0	0
S_3	1	dx	2	sx	0	0
S_4	1	dx	2	dx	1	sx
S_5	1	dx	2	dx	1	dx

Tabella 3.3: Esempio di definizione dei sottogruppi

insiemi fossero meno di 8 si pone uno 0 negli ultimi posti. Alla partizione in Figura 1.1 corrisponderebbe il vettore $(S_1, S_2, S_3, S_4, S_5, 0, 0, 0)$.

Le 8 esplicative considerate in questa analisi generano uno spazio $\mathbf{\Pi}$ di dimensione 2 646 063 e un numero di sottogruppi pari a 4 009. La struttura così impostata, senza la lettura dei dati, richiede una allocazione di memoria pari a 465 MB. Una simulazione che considera un'ipotetica caratteristica binaria aggiuntiva crea 5 408 692 partizioni e 5 563 sottogruppi, per complessivi 950 MB di modello.

Questi numeri considerevoli giustificano la limitazione a 8 variabili e l'impiego del metodo delle distanze descritto nel paragrafo 2.2.2 per la sostituzione delle unità, poiché diversamente il costo mnemonico esorbitante causerebbe l'interruzione della procedura. L'utilizzo di metodi computazionalmente più efficienti è in ogni caso possibile; tuttavia, poiché l'interesse della trattazione è sulla valutazione statistica del metodo proposto, gli aspetti tecnologici sono stati tralasciati.

3.3.2 Allenamento del modello

Costruite le partizioni si procede al calcolo della distribuzione a priori $p(\Pi_r)$ secondo il metodo descritto nel paragrafo 1.5.2. Per quanto riguarda la distribuzione a priori $p(\boldsymbol{\theta}|\Pi_r)$, invece, si è scelta una Beta(a, b), con $f(\theta_{t,m}; a, b) \propto \theta_{t,m}^{a-1}(1-\theta_{t,m})^{b-1}$, imponendo gli iperparametri $a = b = 1$ per ottenere una densità non informativa Uniforme$(0, 1)$.

L'allenamento del modello comincia con l'estrazione di n_{init} clienti. Si rilevano i valori delle esplicative $\boldsymbol{X}^{(n_{init})}$, i canali di contatto impiegati $\boldsymbol{z}^{(n_{init})}$ e le eventuali sottoscrizioni del prodotto $\boldsymbol{y}^{(n_{init})}$, così da ottenere la verosimiglianza $p(\boldsymbol{y}^{(n_{init})}|\boldsymbol{X}^{(n_{init})}, \boldsymbol{z}^{(n_{init})}, \boldsymbol{\theta}, \Pi_r)$. Essa, opportunamente combinata con le distribuzioni a priori, consente il calcolo della distribuzione congiunta a posteriori $p(\boldsymbol{y}^{(n_{init})}, \boldsymbol{\theta}, \Pi_r|\boldsymbol{X}^{(n_{init})}, \boldsymbol{z}^{(n_{init})})$ e della probabilità predittiva a posteriori $q(t, \boldsymbol{x}_{n_{init}+1})$.

Successivamente si effettua un campionamento casuale di ω unità e si effettuano le previsioni sui migliori canali di contatto, sfruttando la probabilità predittiva a posteriori $q(t, \boldsymbol{x}_{n_{init}+1})$. Le unità che nel dataset erano state assegnate alla strategia corretta vengono mantenute, mentre i soggetti con canale errato vengono sostituiti tramite il metodo delle distanze descritto nel paragrafo 2.2.2. La *wave* è quindi composta da un numero $\omega^* \in \{1, ..., \omega\}$ di individui, che vengono aggiunti agli n_{init} iniziali. Si consideri $n_\omega = n_{init} + \omega^*$ la numerosità totale dei soggetti studiati dopo la prima ondata.

A questo punto si calcola nuovamente la verosimiglianza del modello $p(\boldsymbol{y}^{(n_\omega)}|\boldsymbol{X}^{(n_\omega)}, \boldsymbol{z}^{(n_\omega)}, \boldsymbol{\theta}, \Pi_r)$ sfruttando tutti i dati finora analizzati, ottenendo la distribuzione a posteriori $p(\boldsymbol{y}^{(n_\omega)}, \boldsymbol{\theta}, \Pi_r|\boldsymbol{X}^{(n_\omega)}, \boldsymbol{z}^{(n_\omega)})$ e, infine, la probabilità predittiva a posteriori $q(t, \boldsymbol{x}_{n_\omega+1})$.

La procedura prosegue ripetendo il procedimento della prima *wave* fino al raggiungimento di una numerosità N prefissata.

3.3.3 Parametri utilizzati

Il disegno così definito è stato replicato numerose volte, impiegando ogni combinazione possibile dei seguenti valori dei parametri:

- n_{init}, ampiezza del campione iniziale: 500, 1 000, 1 500
- ω, ampiezza della *wave*: 50, 100, 150
- N, ampiezza del campione finale: 1 000, 2 000, ..., 6 000

Questo consente non solo il confronto con un campione casuale semplice, ma anche un'analisi di sensibilità per la ricerca dei parametri ottimali.

I campioni analizzati nel seguito comprendono solamente le unità estratte nella procedura ad ondate, mentre si trascurano le n_{init} osservazioni casuali impiegate per l'inizializzazione del metodo.

Si sottolinea che la numerosità campionaria effettiva N^* non è esattamente pari al valore nominale N, poiché si considera il primo modello che con una *wave* completa supera l'ampiezza desiderata. Le numerosità campionarie effettive sono riportate in Tabella 3.4.

N	n_{init}	$\omega = 50$	$\omega = 100$	$\omega = 150$
1000	500	1040	1073	1113
	1000	1038	1060	1125
	1500	1036	1072	1009
2000	500	2019	2041	2092
	1000	2018	2029	2120
	1500	2013	2037	2013
3000	500	3050	3016	3089
	1000	3012	3091	3107
	1500	3038	3006	3019
4000	500	4034	4087	4081
	1000	4045	4057	4122
	1500	4025	4076	4028
5000	500	5013	5054	5074
	1000	5025	5033	5108
	1500	5016	5046	5031
6000	500	6046	6019	6073
	1000	6008	6092	6091
	1500	6044	6006	6035

Tabella 3.4: Numerosità campionarie effettive

La costruzione dei campioni a diverse ampiezze finali è stata svolta in

maniera successiva: per ogni coppia (n_{init}, ω) è stato allenato il modello fino a $N = 6\,000$, salvando i risultati intermedi ai valori di N sopra indicati. Ciò implica che, fissati (n_{init}, ω), le osservazioni presenti nei campioni con N più piccolo sono presenti anche nei campioni ad ampiezza superiore. Ad esempio, per $(n_{init} = 500, \omega = 50)$, le 1 040 unità statistiche del campione con $N = 1\,000$ fanno parte anche del campione con $N = 2\,000$, $N = 3\,000$, e così via.

3.3.4 Campioni casuali semplici

Ad ogni campione SUBA ottenuto da una combinazione dei parametri è stato affiancato un campione casuale semplice.

Essi sono stati costruiti per avere la medesima ampiezza campionaria effettiva N^* dei rispettivi campioni SUBA, così che non ci sia alcuna differenza di numerosità ad influire sulle analisi.

Inoltre si è scelto il medesimo meccanismo ad espansione: fissata la coppia (n_{init}, ω), il campione con $N = 1\,000$ è stato copiato ed accresciuto fino ad arrivare a $N = 2\,000, 3\,000, ...$ unità. Ciò consente di confrontare equamente le due procedure.

3.4 Risultati

Per ogni combinazione di valori dei parametri è stato ottenuto un vettore con gli indici dei soggetti inseriti nel campione finale. Di questi individui si è calcolata la percentuale di sottoscrizioni dell'offerta promossa p_S, al fine di confrontarla con la proporzione di successi ottenuta nel corrispondente campione casuale semplice p_C. I risultati sono disponibili in Tabella 3.5.

Il confronto statistico viene effettuato sulla base di un test z per proporzioni campionarie indipendenti. Il sistema di ipotesi è:

$$\begin{aligned} H_0 &: p_S = p_C \\ H_1 &: p_S > p_C \end{aligned} \tag{3.1}$$

		Metodo SUBA			Casuale semplice		
N	n_{init}	$\omega = 50$	$\omega = 100$	$\omega = 150$	$\omega = 50$	$\omega = 100$	$\omega = 150$
1000	500	0.1433	0.1174	0.1123	0.1135	0.0923	0.1078
	1000	0.1108	0.1151	0.1253	0.1310	0.1311	0.1333
	1500	0.1264	0.1222	0.1318	0.1197	0.1119	0.1001
2000	500	0.1362	0.1230	0.1119	0.1114	0.1102	0.1061
	1000	0.1115	0.1119	0.1184	0.1120	0.1045	0.1146
	1500	0.1207	0.1163	0.1292	0.1157	0.1124	0.1326
3000	500	0.1298	0.1233	0.1140	0.1118	0.1111	0.1104
	1000	0.1152	0.1129	0.1175	0.1026	0.1113	0.1204
	1500	0.1192	0.1151	0.1312	0.1090	0.1161	0.1033
4000	500	0.1244	0.1199	0.1122	0.1091	0.1126	0.1054
	1000	0.1194	0.1154	0.1167	0.1172	0.1136	0.1189
	1500	0.1180	0.1168	0.1293	0.1125	0.1121	0.1092
5000	500	0.1213	0.1197	0.1147	0.1081	0.1100	0.1090
	1000	0.1160	0.1142	0.1159	0.1144	0.1067	0.1098
	1500	0.1160	0.1140	0.1252	0.1192	0.1058	0.1163
6000	500	0.1196	0.1188	0.1158	0.1080	0.1106	0.1120
	1000	0.1168	0.1152	0.1182	0.1114	0.1146	0.1159
	1500	0.1165	0.1141	0.1243	0.1155	0.1102	0.1140

Tabella 3.5: Proporzioni di successo

Si è scelto di impostare un'alternativa unilaterale perché si suppone che il metodo proposto non abbia risultati peggiori del campione casuale. Infine si assume equivarianza per le due popolazioni, poiché è noto che esse coincidono con il medesimo dataset completo di N_{pop} osservazioni. La statistica test ha dunque la forma seguente:

$$z = \frac{p_S - p_C}{\sqrt{\frac{p_S(1-p_S)+p_C(1-p_C)}{N}}} \tag{3.2}$$

con regione di rifiuto situata sulla coda destra della distribuzione nulla, che coincide asintoticamente con una Normale$(0, 1)$.

Le significatività osservate del test così costruito sono riportate in Tabella 3.6.

N	n_{init}	$\omega = 50$	$\omega = 100$	$\omega = 150$
1000	500	0.0210	0.0284	0.3674
	1000	0.5786	0.6305	0.7859
	1500	0.3198	0.2300	0.0130
2000	500	0.0084	0.1024	0.2758
	1000	0.9801	0.2242	0.3509
	1500	0.3127	0.3469	0.8718
3000	500	0.0153	0.0693	0.3287
	1000	0.0580	0.4201	0.8622
	1500	0.1054	0.9518	0.0004
4000	500	0.0158	0.1502	0.1598
	1000	0.3783	0.4036	0.8792
	1500	0.2213	0.2543	0.0027
5000	500	0.0193	0.0632	0.1805
	1000	0.4013	0.1135	0.1662
	1500	0.8100	0.0959	0.0843
6000	500	0.0225	0.0805	0.2556
	1000	0.1718	0.4548	0.3466
	1500	0.4323	0.2530	0.0407

Tabella 3.6: Significatività osservate

3.4.1 Analisi di sensibilità

In generale, l'analisi congiunta delle proporzioni di successo in Tabella 3.5 e delle significatività osservate in Tabella 3.6 evidenzia che la combinazione

migliore di parametri (n_{init}, ω) è data da $(n_{init} = 500, \omega = 50)$. Infatti, al variare di N questa coppia di valori è quella che manifesta un andamento più distante dal campione casuale semplice con cui viene confrontata. Questo suggerisce che n_{init} e ω dovrebbero essere il più piccoli possibile per ottenere le migliori prestazioni. Le ragioni di questa evidenza sono dettagliate nei prossimi paragrafi.

Ruolo dell'ampiezza del campione iniziale n_{init}

La dimensione n_{init} ha un grande impatto sulle prestazioni del metodo. Il campione iniziale, infatti, è estratto tramite un campionamento casuale semplice e in questa fase non avviene né una selezione ragionata delle unità né il calcolo del canale di contatto ottimale.

Tuttavia, questo campione è necessario a fini esplorativi e di inizializzazione, poiché, tramite il calcolo della verosimiglianza, consente di ottenere la probabilità predittiva a posteriori $q(t, \boldsymbol{x}_{n+1})$ impiegata per l'ottimizzazione del canale di contatto. L'apprendimento in linea del disegno, quindi, inizia dalla formazione della prima ondata di clienti.

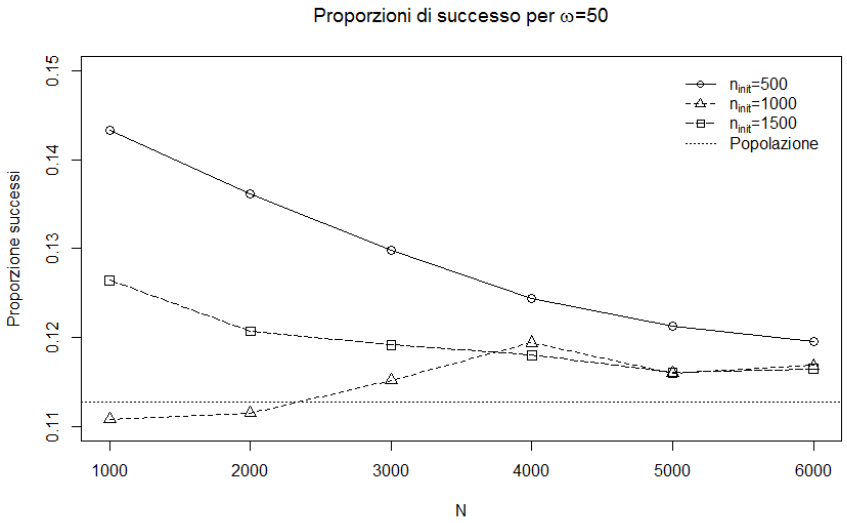


Figura 3.2: Proporzioni di successo per $\omega = 50$

Nella simulazione del disegno si ipotizzava che un campione iniziale più ampio consentisse un più rapido apprendimento, ma i risultati non ne danno evidenza. Infatti, a parità di ω si osserva come l'incremento di n_{init} porti il campione SUBA ad assomigliare sempre più al campione casuale semplice. Questo andamento è spiegato dal meccanismo della sostituzione delle unità. Si ricorda che l'applicazione del metodo a dati già raccolti consente solamente di prevedere il canale di contatto ottimale per ogni soggetto, ma non di assegnarlo: individui sottoposti alla strategia errata vengono perciò sostituiti.

All'aumentare di n_{init} il numero di osservazioni disponibili per la sostituzione nelle ondate si riduce, dunque la densità dei punti disponibili nello spazio delle variabili è sempre più piccola e la distanza da coprire per effettuare lo scambio è sempre più ampia. Il campione iniziale, quindi, è necessario per inizializzare il metodo, ma non deve essere troppo numeroso per non influire negativamente sulla fase successiva.

In particolare, i modelli che meglio dimostrano quanto affermato sono quelli con $\omega = 50$ e $\omega = 100$, poiché, fissato ω, si rileva che le proporzioni di successo ottenute con $n_{init} = 500$ sono più elevate rispetto ai campioni con valori differenti di n_{init}.

In Figura 3.2 si possono osservare le proporzioni di successo per campioni con $\omega = 50$ e diversi valori di n_{init}: risulta evidente che il modello sviluppato con $n_{init} = 500$ è quello che si discosta maggiormente dalla media della popolazione. Inoltre i modelli con $n_{init} = 1\,000$ e $n_{init} = 1\,500$, già meno efficienti, si dimostrano equivalenti per numerosità finale $N \geq 4\,000$.

Ruolo dell'ampiezza della *wave* ω

Per quanto riguarda l'ampiezza ω dell'ondata si rileva un effetto simile a quanto rilevato per n_{init}: al diminuire di ω aumentano le prestazioni del disegno.

Ciò avviene perché più corta è la *wave*, maggiore è il numero delle *waves* stesse: di conseguenza la verosimiglianza viene ricalcolata un numero superiore di volte e l'apprendimento è più veloce. A tal proposito si ricorda che il calcolo della funzione di probabilità predittiva a posteriori $q(t, \boldsymbol{x}_{n+1})$

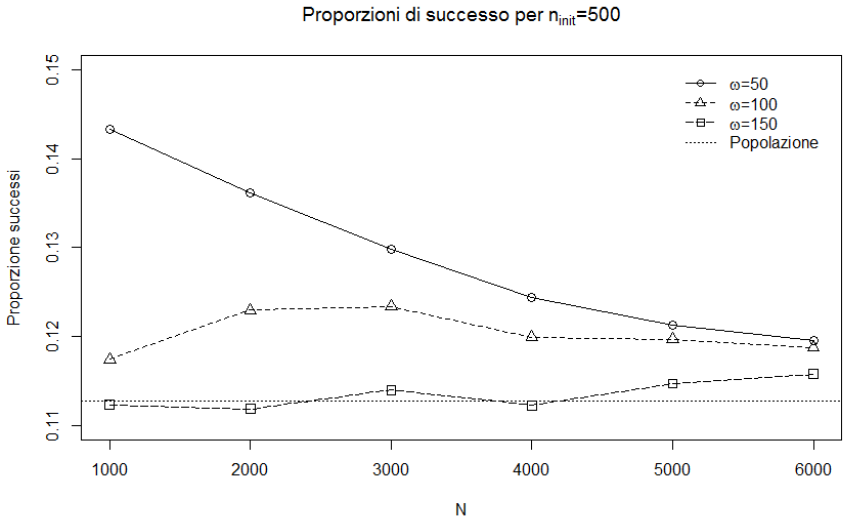


Figura 3.3: Proporzioni di successo per $n_{init} = 500$

avviene al termine di ogni ondata completa, quindi solo un numero maggiore di ondate può portare ad un aggiornamento più frequente.

In Tabella 3.7 viene riportato il numero di ondate necessarie alla costruzione di ogni campione. Si nota che il modello con ω nominale pari a 50 ha mediamente il doppio degli aggiornamenti del caso con $\omega = 100$ e il triplo di $\omega = 150$: esso è dunque più preciso ed efficiente nella selezione dei soggetti con risposta positiva, come si osserva anche in Figura 3.3.

Ruolo dell'ampiezza finale N

L'effetto della dimensione del campione finale è invece piuttosto controverso. Se intuitivamente si potrebbe affermare che all'aumentare di N la proporzione di successi cresca, un'analisi più approfondita rivela che l'andamento è più complesso.

In Figura 3.4 si esamina con dettaglio il campione ottenuto con ($n_{init} = 500, \omega = 50$), che presenta i più elevati livelli di significatività. Sono rappresentate le proporzioni di successo delle singole ondate e la proporzione cumulata in funzione dell'ampiezza campionaria N, utilizzando come riferimento la vera percentuale di successo nella popolazione. Si osserva come fino

N	n_{init}	$\omega = 50$	$\omega = 100$	$\omega = 150$
1000	500	21	11	8
	1000	21	11	8
	1500	21	11	7
2000	500	41	21	15
	1000	41	21	15
	1500	41	21	14
3000	500	62	31	22
	1000	61	32	22
	1500	62	31	21
4000	500	82	42	29
	1000	82	42	29
	1500	82	42	28
5000	500	102	52	36
	1000	102	52	36
	1500	102	52	35
6000	500	123	62	43
	1000	123	63	43
	1500	123	62	42

Tabella 3.7: Numero di *waves*

a 1 500 osservazioni le ondate abbiano una percentuale di successi generalmente maggiore del valore rilevato nella popolazione, con oscillazioni di poco conto al di sotto del riferimento. Inoltre si nota che la proporzione cumulata è tendenzialmente stabile fino a $N = 2\,500$ per poi decrescere lentamente, pur mantenendosi significativamente superiore alla media della popolazione.

Anche in questo caso la motivazione principale è da cercare nella sostituzione delle osservazioni. Il procedimento, infatti, è tanto più efficace quanto più la numerosità campionaria N è piccola in rapporto a quella della popolazione N_{pop}, perché sono più numerose le unità disponibili per la

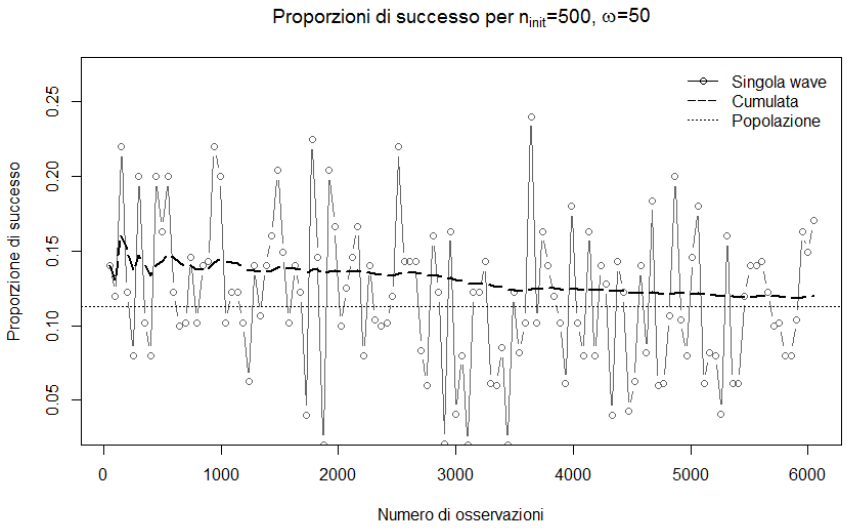


Figura 3.4: Proporzioni di successo per $n_{init} = 500$ e $\omega = 50$

sostituzione.

Variabilità delle *waves*

La Figura 3.4 evidenzia anche una grande variabilità della percentuale di successi nella singola ondata. Inizialmente ciò risulta sorprendente, poiché ci si attendeva un costante aumento delle proporzioni di successo in ogni *wave*, con poche variazioni intorno alla percentuale cumulata.

Si osserva come fino a 1 700 osservazioni la variabilità sia inferiore rispetto al resto della serie. Da 1 700 in poi, ondate con risultati decisamente al di sopra del valore della popolazione sono seguite da altre con proporzioni nettamente inferiori.

La variabilità nel suo complesso, tuttavia, è dovuta alla numerosità ω delle ondate, come si può vedere in Figura 3.5. Come riportano Pace & Salvan (2001), infatti, la varianza della proporzione campionaria è inversamente proporzionale alla numerosità del campione. Ciò significa che la varianza dello stimatore per $\omega = 50$ è doppia rispetto al caso con $\omega = 100$ e tripla rispetto a $\omega = 150$.

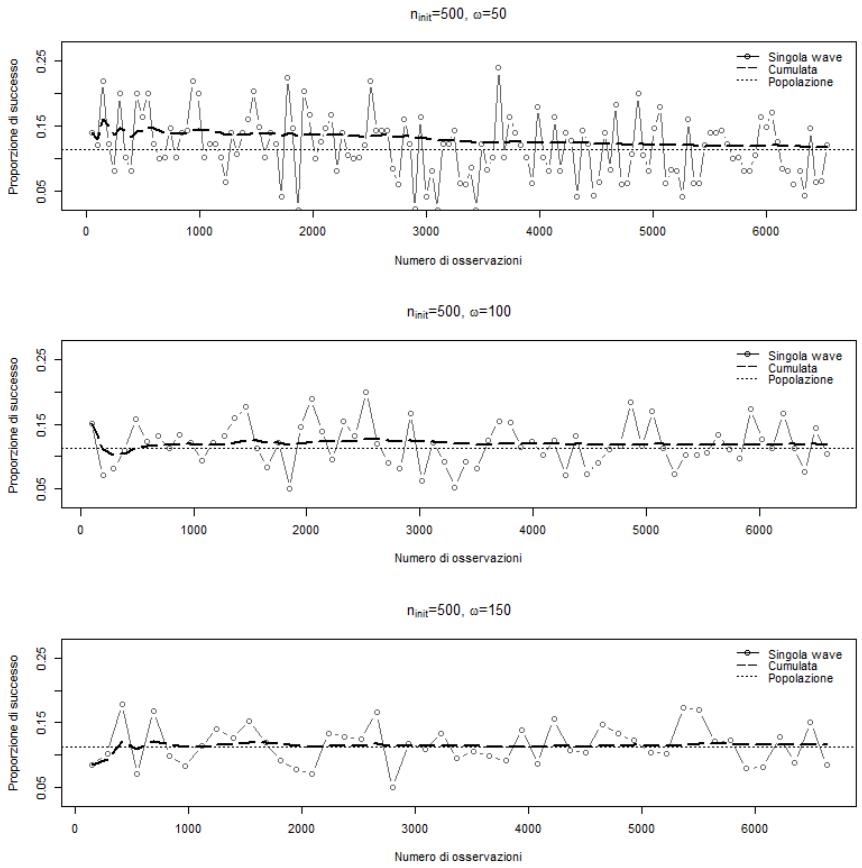


Figura 3.5: Confronto di variabilità tra *waves* di ampiezza differente

La riduzione della variabilità, quindi, si ottiene solamente aumentando il valore ω. Dall'altra parte, però, poiché un incremento dell'ampiezza dell'ondata causa un peggioramento nella capacità di evidenziare i successi, il controllo della variabilità delle *waves* passa in secondo piano.

3.5 Considerazioni finali

Riassumendo, si suggerisce di utilizzare n_{init} e ω più piccoli possibile, per consentire un miglior apprendimento del modello.

Ciò è coerente con i risultati ottenuti da Xu et al. (2016) che hanno impostato il disegno originale per l'utilizzo di $\omega = 1$. Si consideri che in

ambito biomedico l'esperimento è guidato anzitutto da principi etici, per cui il metodo deve essere in grado di imparare velocemente per attribuire la miglior cura ad ogni paziente. Inoltre le osservazioni disponibili sono sempre poche, con la conseguenza che la conduzione completa dello studio con ondate unitarie ha una durata non eccessivamente lunga.

In questo esempio, invece, si è preferito concentrarsi su ω più elevati poiché si era interessati ad una simulazione realistica in campo aziendale. La costruzione di *waves* con $\omega = 1$ è possibile per l'impiego su dati già raccolti, ma si rivela poco realizzabile qualora si decidesse di applicare il medesimo disegno su dati in linea.

Si è quindi esplorata la possibilità che la simulazione possa non solo portare risultati di per sé, esplicitando una segmentazione della clientela tramite un'enorme quantità di dati già disponibili, ma che sia anche punto di partenza per futuri esperimenti su nuovi dati. In quest'ultimo caso, l'ondata unitaria è poco sostenibile in termini di tempo e costi, poiché è necessario aspettare la risposta del soggetto prima di contattare il successivo, con un'attesa che può rivelarsi anche molto lunga a seconda del canale utilizzato per il contatto.

Dal punto di vista numerico, la simulazione qui riportata rivela che $n_{init} \approx \frac{N_{pop}}{100}$ e $\omega \approx \frac{N_{pop}}{1000}$ potrebbero costituire i massimi valori accettabili per ottenere risultati interessanti.

Per quanto riguarda il valore di N, invece, si propone un limite pari a $N \approx \frac{N_{pop}}{10}$. Come si è dimostrato, un valore troppo elevato di N comporta una riduzione delle prestazioni del disegno a causa del metodo di sostituzione delle unità. L'incremento di N riduce il numero di osservazioni disponibili per lo scambio e con esso la loro densità nello spazio delle variabili. Ciò implica che è necessario coprire distanze maggiori per effettuare lo scambio, per cui unità considerate "lontane" con N piccolo diventano "vicine" con N grande in mancanza di osservazioni più prossime. La sostituzione, quindi, diventa sempre meno precisa all'aumentare di N.

Il problema è strettamente legato alla maledizione della dimensionalità, secondo cui l'incremento del numero di variabili deve essere accompagnato da un aumento esponenziale delle unità, come descritto in dettaglio da Ha-

stie et al. (2009). Per queste ragioni si suggerisce di applicare il metodo proposto a dataset molto grandi, che consentono cioè un'adeguata densità di osservazioni.

Le valutazioni qui condotte sono guidate esclusivamente dalla percentuale di successi catturata con la selezione del campione. È possibile ottenere anche la descrizione della partizione ottimale per l'analisi della segmentazione della clientela, tuttavia questo procedimento non è inerente alla validazione del metodo proposto ed è stato tralasciato.

Infine si segnala che dal punto di vista empirico, il tempo di svolgimento delle simulazioni non sembra risentire molto dell'impostazione dei parametri, sebbene valori di ω più elevati abbiano richiesto tempo di calcolo generalmente più lungo.

Conclusioni

Il disegno adattivo basato sui sottogruppi (*Subgroup-Based Adaptive design* – SUBA) proposto da Xu et al. (2016) si è dimostrato un metodo versatile ed applicabile non solo in campo biomedico, ma anche nella promozione aziendale. Infatti se in biostatistica l'interesse era rivolto al calcolo del miglior trattamento per ottenere la guarigione del soggetto, nel marketing è possibile sfruttare questo disegno per ottimizzare il canale di contatto pubblicitario per ogni cliente, così da ottenere il massimo numero di adesioni alla campagna.

Al metodo così costruito è stata aggiunta la potenzialità di un impiego su dati già raccolti, al fine di ottenere una segmentazione della clientela. Questo adattamento possiede il vantaggio di poter estrarre in maniera intelligente gli individui che hanno risposto favorevolmente a precedenti campagne pubblicitarie per studiarne meglio le caratteristiche.

Il caso studio presentato dimostra l'applicabilità del metodo proposto. Dai risultati è emerso infatti che il campione estratto dal modello è significativamente diverso da un campione casuale semplice con riferimento alla percentuale di successo.

Un'analisi dettagliata dei parametri utilizzati suggerisce che è preferibile un insieme iniziale di piccole dimensioni, affinché inizializzi il disegno con poche unità casuali, lasciando un numero maggiore di soggetti per la fase successiva. Inoltre si è visto che ondate con poche unità hanno prestazioni migliori, poiché consentono un aggiornamento più frequente del modello e dunque un apprendimento più veloce.

Il metodo risulta tanto migliore quanto maggiore è il numero di osservazioni del dataset complessivo, ma in azienda si supera facilmente il milione di unità, valore che sicuramente consente l'applicazione accurata del disegno.

Il caso trattato è solo un esempio, con pochi riferimenti al contesto specifico, di quanto è possibile realizzare in una realtà aziendale. Certamente la conoscenza dell'ambiente in cui sono stati prodotti i dati non può che essere un vantaggio a favore delle prestazioni del metodo. Questo renderebbe anche possibile una selezione ragionata del campione iniziale, costruito con osservazioni rispondenti ad un modello già esistente di assegnazione dei canali di contatto invece che con estrazioni casuali, e dovrebbe consentire un apprendimento più veloce del disegno.

Infine, nei prossimi anni le capacità di elaborazione e le disponibilità di memoria dei calcolatori sono destinate a crescere con sempre maggiore velocità. Sarà quindi possibile effettuare studi sempre più onerosi dal punto di vista computazionale, superando barriere che non sono state affrontate nell'analisi presente. Potrebbero essere svolte molte più simulazioni per ottenere raccomandazioni più precise sul valore ottimale dei parametri e nuove applicazioni del disegno potrebbero prendere in considerazione variabili categoriali con tre o più livelli, spesso presenti nei database aziendali.

Appendice A

Codice R

A.1 Pacchetti utilizzati

Per le analisi è stato utilizzato il software R versione 3.5.1 (*Feather Spray*) o 3.4.3 (*Kite-Eating Tree*) in caso di pacchetti non ancora aggiornati (R Core Team (2018)). Inoltre sono stati impiegati i seguenti pacchetti per l'implementazione dl metodo:

- `doParallel`, Microsoft Corporation & Weston (2017*a*)
- `doSNOW`, Microsoft Corporation & Weston (2017*b*)
- `foreach`, Microsoft Corporation & Weston (2017*c*)
- `hopach`, van der Laan & Pollard (2003)

ed altri per la selezione delle variabili e i confronti con il campione casuale semplice:

- `caret`, Kuhn (2018)
- `plyr`, Wickham (2011)
- `pROC`, Robin et al. (2011)
- `earth`, Milborrow (2018)
- `nnet`, Venables & Ripley (2002)
- `randomForest`, Liaw & Wiener (2002)

A.2 Funzioni R

```
# EMPTY MODEL BUILDING ----

SUBA_partition <- function(num_vars,is_binary){

  print("Partitions: expanding grid...")
  part <- expand.grid(rep(list(1:(num_vars+1)),7))
  print("Partitions: grid expanded!")

  # Removing duplicates: first round ----
  print("Partitions: removing duplicates. First round...
     ")
  part[part[,1] == (num_vars+1),2:7] <- num_vars+1
  part[part[,2] == (num_vars+1),4:5] <- num_vars+1
  part[part[,3] == (num_vars+1),6:7] <- num_vars+1
  part <- unique(part)
  part <- as.matrix(part[do.call(order,as.data.frame(
     part)) ,])
  print("Partitions: first round finished!")

  R <- dim(part)[1]

  # Managing binary variables ----
  #They can be splitted at most once
  print("Partitions: managing binary variables...")
  for(k in 1:num_vars){
    if(is_binary[k]){
      index1 <- (1:R)[part[,1] == k]
      part1 <- part[index1,2:7]
      part1[part1 == k] <- num_vars+1
      part[index1,2:7] <- part1

      index2 <- (1:R)[part[,2] == k]
      part2 <- part[index2,4:5]
      part2[part2 == k] <- num_vars+1
      part[index2,4:5] <- part2

      index3 <- (1:R)[part[,3] == k]
      part3 <- part[index3,6:7]
      part3[part3 == k] <- num_vars+1
      part[index3,6:7] <- part3
    }
  }

  # Removing duplicates: second round ----
  print("Partitions: removing duplicates. Second
     round...")
  part[part[,1] == (num_vars+1),2:7] <- num_vars+1
```

```
  part[part[,2] == (num_vars+1),4:5] <- num_vars+1
  part[part[,3] == (num_vars+1),6:7] <- num_vars+1
  part <- unique(part)
  part <- as.matrix(part[do.call(order,as.data.frame(
     part)) ,])
  print("Partitions: all duplicates removed!")

  R <- dim(part)[1] #number of all possible partitions
  part_prior <- rep(0,R)

  # Priors of all partitons ----
  for(r in 1:R){
    if ((r %% 500) == 0 | r == R) print(sprintf("
       Computing partition priors: partition %d out of %
       d...",r,R))

    i1 <- part[r,1];i2 <- part[r,2];i3 <- part[r,3];i4
       <- part[r,4];i5 <- part[r,5];i6 <- part[r,6];i7
       <- part[r,7]

    HB <- sort(part[r,1:7])
    HA <- sum(HB<=num_vars)
    HC <- 1+sum(HB[1:6]<HB[2:7])-(sum(HB == (num_vars+1)
       )>0) #count of dimension used for splitting

    #hyperparameters
    pr_nosplit <- 1/(num_vars+1)
    pr_split <- 1/(num_vars+1)
    split_phi <- 0.5

    #computing
    if(i1 == (num_vars+1)){
      pr <- pr_nosplit^3
    }
    if(i1<=num_vars & i2 == (num_vars+1) & i3 == (
       num_vars+1)){
      pr <- pr_split*pr_nosplit^2*pr_nosplit^2
    }
    if(i1<=num_vars & i2 == (num_vars+1) & i3<=num_vars
       & i6 == (num_vars+1) & i7 == (num_vars+1)){
      pr <- pr_split*pr_nosplit^2*pr_split*pr_nosplit^2
    }
    if(i1<=num_vars & i2 == (num_vars+1) & i3<=num_vars
       & i6 == (num_vars+1) & i7<=num_vars){
      pr <- pr_split*pr_nosplit^2*pr_split*pr_nosplit*
         pr_split*(split_phi)^(HC-1)
    }
    if(i1<=num_vars & i2 == (num_vars+1) & i3<=num_vars
       & i6<=num_vars & i7 == (num_vars+1)){
```

```
      pr <- pr_split*pr_nosplit^2*pr_split*pr_nosplit*
         pr_split*(split_phi)^(HC-1)
    }
    if(i1<=num_vars & i2 == (num_vars+1) & i3<=num_vars
       & i6<=num_vars & i7<=num_vars){
      pr <- pr_split*pr_nosplit^2*pr_split*pr_split*
         pr_split*(split_phi)^(HC-1)
    }
    if(i1<=num_vars & i2<=num_vars & i3 == (num_vars+1)
       & i4 == (num_vars+1) & i5 == (num_vars+1)){
      pr <- pr_split*pr_nosplit^2*pr_split*pr_nosplit^2
    }
    if(i1<=num_vars & i2<=num_vars & i3 == (num_vars+1)
       & i4 == (num_vars+1) & i5<=num_vars){
      pr <- pr_split*pr_nosplit^2*pr_split*pr_nosplit*
         pr_split*(split_phi)^(HC-1)
    }
    if(i1<=num_vars & i2<=num_vars & i3 == (num_vars+1)
       & i4<=num_vars & i5 == (num_vars+1)){
      pr <- pr_split*pr_nosplit^2*pr_split*pr_nosplit*
         pr_split*(split_phi)^(HC-1)
    }
    if(i1<=num_vars & i2<=num_vars & i3 == (num_vars+1)
       & i4<=num_vars & i5<=num_vars){
      pr <- pr_split*pr_nosplit^2*pr_split*pr_split*
         pr_split*(split_phi)^(HC-1)
    }
    if(i1<=num_vars & i2<=num_vars & i3<=num_vars) {
      pr <- pr_split^3*pr_nosplit^(7-HA)*pr_split^(HA-3)
         *(split_phi)^(HC-1)
    }

    part_prior[r] <- pr
  }

  # Normalizing prior ----
  part_prior <- part_prior / sum(part_prior)

  return(list(part=part,part_prior=part_prior))
  # part: large matrix (R*7), each column correspond to
     a node in the split tree
  # part_prior: vector (R) with the prior
}


SUBA_subgroup <- function(num_vars,is_binary){

  print("Subgroups: expanding grid...")
  subgroup <- expand.grid(rep(list(1:(num_vars+1),0:1)
```

```
    ,3))
  print("Subgroups: grid expanded!")

  # Removing duplicates: first round ----
  print("Subgroups: removing duplicates. First round..."
    )
  subgroup[subgroup[,1] == (num_vars+1),2:6] <- num_vars
    +1
  subgroup[subgroup[,3] == (num_vars+1),4:6] <- num_vars
    +1
  subgroup[subgroup[,5] == (num_vars+1),6] <- num_vars+1
  subgroup <- unique(subgroup)
  subgroup <- as.matrix(subgroup[do.call(order,
    as.data.frame(subgroup)) ,])
  print("Subgroups: first round finished!")

  M <- dim(subgroup)[1]

  # Managing binary variables ----
  #They can be splitted at most once
  print("Subgroups: managing binary variables...")
  for(k in 1:num_vars){
    if(is_binary[k]){

      index1 <- (1:M)[subgroup[,1] == k]
      subgroup1 <- subgroup[index1,c(3,5)]
      subgroup1[subgroup1 == k] <- num_vars+1
      subgroup[index1,c(3,5)] <- subgroup1

      index2 <- (1:M)[subgroup[,3] == k]
      subgroup2 <- subgroup[index2,5]
      subgroup2[subgroup2 == k] <- num_vars+1
      subgroup[index2,5] <- subgroup2
    }
  }

  # Removing duplicates: second round ----
  print("Subgroups: removing duplicates. Second round...
    ")
  subgroup[subgroup[,1] == (num_vars+1),2:6] <- num_vars
    +1
  subgroup[subgroup[,3] == (num_vars+1),4:6] <- num_vars
    +1
  subgroup[subgroup[,5] == (num_vars+1),6] <- num_vars+1
  subgroup <- unique(subgroup)
  subgroup <- as.matrix(subgroup[do.call(order,
    as.data.frame(subgroup)) ,])
  print("Subgroups: all duplicates removed!")
```

```
  return(subgroup)
  # subgroup: row-named matrix (M*6) with the path (var,
     right/left;var,right/left;var,right/left) of each
     subset (leaf of the tree)
}


SUBA_partsub <- function(part,subgroup,num_vars){

  R <- dim(part)[1] #number of all possible partitions
  partsub <- matrix(0,R,8)

  for(r in 1:R){
    if ((r %% 500) == 0 | r == R) print(sprintf("Finding
       partition subgroups: partition %d out of %d...",
      r,R))

    # Initializing
    i1 <- part[r,1];i2 <- part[r,2];i3 <- part[r,3];i4
       <- part[r,4];i5 <- part[r,5];i6 <- part[r,6];i7
       <- part[r,7]

    # Assegning subgroups ----
    if (i1 == (num_vars+1)){
      partsub[r,1] <- which(subgroup[,1] == i1)
    }

    if (i1 < (num_vars+1) & i2 == (num_vars+1) & i3 == (
       num_vars+1)){
      partsub[r,1] <- which(subgroup[,1] == i1 &
         subgroup[,2] == 0 & subgroup[,3] == i2)
      partsub[r,2] <- which(subgroup[,1] == i1 &
         subgroup[,2] == 1 & subgroup[,3] == i3)
    }

    if (i2 < (num_vars+1) & i3 < (num_vars+1)){
      partsub[r,1:2] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 0 &  subgroup[,3]  == i2 &
         subgroup[,4]  == 0 &  subgroup[,5]  == i4)
      partsub[r,3:4] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 0 &  subgroup[,3]  == i2 &
         subgroup[,4]  == 1 &  subgroup[,5]  == i5)
      partsub[r,5:6] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 1 &  subgroup[,3]  == i3 &
         subgroup[,4]  == 0 &  subgroup[,5]  == i6)
      partsub[r,7:8] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 1 &  subgroup[,3]  == i3 &
         subgroup[,4]  == 1 &  subgroup[,5]  == i7)
    }
```

```
    if (i2 < (num_vars+1) & i3 == (num_vars+1)){
      partsub[r,1:2] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 0 &  subgroup[,3]  == i2 &
         subgroup[,4]  == 0 &  subgroup[,5]  == i4)
      partsub[r,3:4] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 0 &  subgroup[,3]  == i2 &
         subgroup[,4]  == 1 &  subgroup[,5]  == i5)
      partsub[r,5] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 1 &  subgroup[,3]  == i3)
    }

    if (i2 == (num_vars+1) & i3 < (num_vars+1)){
      partsub[r,1] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 0 &  subgroup[,3]  == i2)
      partsub[r,2:3] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 1 &  subgroup[,3]  == i3 &
         subgroup[,4]  == 0 &  subgroup[,5]  == i6)
      partsub[r,4:5] <- which(subgroup[,1] == i1 &
         subgroup[,2]  == 1 &  subgroup[,3]  == i3 &
         subgroup[,4]  == 1 &  subgroup[,5]  == i7)
    }
  }

  # Removing duplicates ----
  #collecting all subsets in the first part of the line
     and put 0 at the (eventual) empty end
  print("Partition subgroups: removing duplicates...")
  unique_partsub <- matrix(0,R,8)
  for(r in 1:R){
    ul <- unique(partsub[r,])
    unique_partsub[r,1:length(ul)] <- ul
  }
  partsub <- unique_partsub
  print("Partition subgroups: all duplicates removed!")

  return(partsub)
  #partsub: R*8 matrix, in each line it contains the
     name of the 1-8 subgroups corresponding to
     partition r
}


SUBA_Xgrid <- function(num_vars,H0=10){

  print("Early stop: grid building...")
  xgrid_perdim <- seq(-1,1,length.out=H0)
  Xgrid <- as.matrix(expand.grid(rep(list(xgrid_perdim),
     num_vars)))
```

```
  print("Early stop: grid built!")

  return(Xgrid)
  #Xgrid: matrix with the cohordinate of points used for
      early stop evaluation
}


# TRAINING ----

SUBA_partition_likelihood <- function(X,Y,Z,SUBA_trained
   ){

  # X: variables dim (N*num_vars)
  # Y: response dim (N)
  # Z: treatment dim (N)

  # Initializing ----
  part <- SUBA_trained$part
  subgroup <- SUBA_trained$subgroup
  partsub <- SUBA_trained$partsub
  num_vars <- SUBA_trained$num_vars
  num_treatm <- SUBA_trained$num_treatm
  is_binary <- SUBA_trained$is_binary
  hp <- SUBA_trained$hp

  # hyperparameters
  a <- hp$a
  b <- hp$b

  M <- dim(subgroup)[1] #number of all possible
     subgroups

  subgroup_loglik <- rep(0,M) #log likelihood by
     subgroup
  subgroup_pprr <- matrix(a/(a+b),M,num_treatm) #
     posterior predictive response rate of treatment t
     by subgroup m

  subgroup_split_point <- matrix(0,M,6)
  subgroup_split_point[,c(1,3,5)] <- subgroup[,c(1,3,5)]
      #split points for all subgroups

  R <- dim(part)[1] #number of all possible partitions

  # Computing subgroup log-likelihood ----
  for(m in 1:M){

    # Finding the data corresponding to the subgroup m
```

```
    ----
  X_sub <- X
  Y_sub <- Y
  Z_sub <- Z

  for(split_round in c(1,3,5)){
    if(subgroup[m,split_round] <= num_vars){
      split_dimension <- subgroup[m,split_round]

      if(is_binary[split_dimension]){
        split_point <- 0.5
      } else {
        split_point <- median(X_sub[,split_dimension])
      }

      subgroup_split_point[m,split_round+1] <-
         split_point

      # Selecting subgroup observations for next split
          step ----
      index_leq <- (X_sub[,split_dimension] <=
         split_point)
      if(subgroup[m,split_round+1] == 0){ #left arm of
          the split
        X_sub <- X_sub[index_leq,]
        Y_sub <- Y_sub[index_leq]
        Z_sub <- Z_sub[index_leq]
      } else{ #right arm of the split
        X_sub <- X_sub[!index_leq,]
        Y_sub <- Y_sub[!index_leq]
        Z_sub <- Z_sub[!index_leq]
      }

      if(length(X_sub) == num_vars) X_sub <- matrix(
         X_sub,1,num_vars)
    }
  }

  # Computing log-likelihood ----
  if(length(Y_sub) > 0){
    for(t in 1:num_treatm){
      n_mt1 <- sum( (Z_sub == t) & (Y_sub == 1) )
      n_mt0 <- sum( (Z_sub == t) & (Y_sub == 0) )
      subgroup_loglik[m] <- subgroup_loglik[m]+lbeta(a
         +n_mt1,b+n_mt0) - lbeta(a,b)

      subgroup_pprr[m,t] <- (a+n_mt1) / (a+b+n_mt0+
         n_mt1)
    }
```

```
    }
  } # end for(m)

  # Computing partition log-likelihood ----
  part_loglik <- rep(0,R)
  for(r in 1:R){
    # find indexes of all subgroups corresponding to
       partition r
    partsub_r <- partsub[r,]
    part_loglik[r] <- sum(subgroup_loglik[partsub_r])

  } # end for(r)

  # Computing partition likelihood ----
  part_loglik <- part_loglik - max(part_loglik)
  part_lik <- exp(part_loglik)
  part_lik <- part_lik / sum(part_lik)

  return(list(part_lik=part_lik,subgroup_split_point=
     subgroup_split_point,subgroup_pprr=subgroup_pprr))
  # part_lik: vector (R) with the likelihood
  # subgroup_split_point: matrix (M*6) with the split
     points of each subgroup
  # subgroup_pprr: matrix (M*3) with the posterior
     predictive probability of each treatment in
     subgroup
}


SUBA_calc_pprr <- function(x_new,SUBA_trained){
  part <- SUBA_trained$part
  part_prior <- SUBA_trained$part_prior
  subgroup <- SUBA_trained$subgroup
  partsub <- SUBA_trained$partsub
  num_vars <- SUBA_trained$num_vars
  num_treatm <- SUBA_trained$num_treatm
  is_binary <- SUBA_trained$is_binary
  hp <- SUBA_trained$hp

  part_lik <- SUBA_trained$part_lik
  subgroup_split_point <- SUBA_trained$
     subgroup_split_point
  subgroup_pprr <- SUBA_trained$subgroup_pprr
  hp <- SUBA_trained$hp

  # hyperparameters
  a <- hp$a
  b <- hp$b
```

```
  M <- dim(subgroup)[1] #number of all possible
     subgroups
  R <- dim(part)[1] #number of all possible partitions

356 N_new <- dim(x_new)[1] # wave size: number of new
     observations to be allocated
  part_pprr <- array(a/(a+b),c(N_new,R,num_treatm))  #
     posterior predictive response rate of treatment t
     by partition r

  is_x_new_in_subgroup <- matrix(TRUE,N_new,M)
  for(m in 1:M){
361   for(split_round in c(1,3,5)){
        if(subgroup[m,split_round] <= num_vars){

          split_dimension <- subgroup[m,split_round]
          split_point <- subgroup_split_point[m,
             split_round+1]
366
          index_leq_new <- (x_new[,split_dimension] <=
             split_point)

          if(subgroup[m,split_round+1] == 0){
            is_x_new_in_subgroup[!index_leq_new,m] <-
               FALSE
371
          } else{
            is_x_new_in_subgroup[index_leq_new,m] <- FALSE
          }
        }
376   }
  } # end for(m)

  for(r in 1:R){
    # find indexes of all subgroups corresponding to
       partition r
381   partsub_r <- partsub[r,]

    for(i_new in 1:N_new){

      # partsub_r[is_x_new_in_subgroup[i_new,partsub_r]]
         returns the index of subgroup that x_new
         belongs to
386   # subgroup_pprr[...,] then returns the posterior
         predictive response rate of num_treatm
         treatments of that subgroup
      part_pprr[i_new,r,] <- subgroup_pprr[partsub_r[
         is_x_new_in_subgroup[i_new,partsub_r]],]
    }
```

```
  } # end for(r)

  part_post <- part_lik*part_prior #posterior of each
     partition

  pprr_x_new <- apply(part_pprr,c(1,3),function(x){ sum(
     x*part_post) })
  pprr_x_new <- matrix(pprr_x_new,N_new,num_treatm)

  return(pprr_x_new)
  # pprr_x_new[i_new,t] is the posterior response rate
     of treatment t for new observation i_new
}


SUBA_early_stop <- function(SUBA_trained,N_parallel=200)
   {

  treatment <- SUBA_trained$treatment
  part <- SUBA_trained$part
  subgroup <- SUBA_trained$subgroup
  partsub <- SUBA_trained$partsub
  part_prior <- SUBA_trained$part_prior
  num_vars <- SUBA_trained$num_vars
  num_treatm <- SUBA_trained$num_treatm
  is_binary <- SUBA_trained$is_binary
  hp <- SUBA_trained$hp
  Xgrid <- SUBA_trained$Xgrid
  t_dropped <- SUBA_trained$t_dropped

  if(num_treatm != 3){
    stop("Can only handle num_treatm=3 treatments")
  }

  N_grid <- dim(Xgrid)[1]

  # define the boundaries
  xperdim_extra <- c(-1,1)
  Xgrid_extra <- as.matrix(expand.grid(rep(list(c(-1,1))
     ,num_vars)))
  N_grid_extra <- dim(Xgrid_extra)[1]

  if(length(treatment) == 3){

    flag1 <- flag2 <- flag3 <- TRUE

    # first examine the boundaries
    comp0 <- foreach(i0=1:N_grid_extra,.combine=rbind) %
```

```
     dopar% {
    com0 <- rep(0,num_treatm)
    x_new <- matrix(Xgrid_extra[i0,],1,num_vars)
    pprr_x_grid_i0 <- SUBA_calc_pprr(x_new,
       SUBA_trained)
    pprr_x_grid_i0 <- c(pprr_x_grid_i0)
    # print(pprr_x_grid_i0)
    if (pprr_x_grid_i0[1] > pprr_x_grid_i0[2]) com0[1]
        <- 1
    if (pprr_x_grid_i0[2] > pprr_x_grid_i0[3]) com0[2]
        <- 1
    if (pprr_x_grid_i0[3] > pprr_x_grid_i0[1]) com0[3]
        <- 1
    com0
}

sum_comp0 <- colSums(comp0)
if (sum_comp0[1] != 0 & sum_comp0[1] != N_grid_extra
   ) flag1 <- FALSE
if (sum_comp0[2] != 0 & sum_comp0[2] != N_grid_extra
   ) flag2 <- FALSE
if (sum_comp0[3] != 0 & sum_comp0[3] != N_grid_extra
   ) flag3 <- FALSE

# Next,sequentially examine the grid values

for (i1 in 1:round(N_grid / N_parallel)){
  if (flag1 | flag2 | flag3){

    comp <- foreach(i2=1:N_parallel,.combine=rbind)
       %dopar% {
      com <- rep(0,num_treatm)
      i3 <- N_parallel * (i1-1)+i2
      x_new <- matrix(Xgrid[i3,],1,num_vars)
      pprr_x_grid_i3 <- SUBA_calc_pprr(x_new,
         SUBA_trained)
      pprr_x_grid_i3 <- c(pprr_x_grid_i3)
      # print(pprr_x_grid_i3)
      if (pprr_x_grid_i3[1] > pprr_x_grid_i3[2]) com
         [1] <- 1
      if (pprr_x_grid_i3[2] > pprr_x_grid_i3[3]) com
         [2] <- 1
      if (pprr_x_grid_i3[3] > pprr_x_grid_i3[1]) com
         [3] <- 1
      com
    }

    sum_comp <- colSums(comp)
    if (sum_comp[1]/N_parallel != sum_comp0[1]/
```

```
            N_grid_extra) flag1 <- FALSE
          if (sum_comp[2]/N_parallel != sum_comp0[2]/
            N_grid_extra) flag2 <- FALSE
          if (sum_comp[3]/N_parallel != sum_comp0[3]/
            N_grid_extra) flag3 <- FALSE
        } else break
      }

      # the index for the treatment that is going to be
        dropped (if any)
      t_drop <- c()
      if (flag1)
        t_drop <- c(t_drop,ifelse(sum_comp[1] == 0,1,2))
      if (flag2)
        t_drop <- c(t_drop,ifelse(sum_comp[2] == 0,2,3))
      if (flag3)
        t_drop <- c(t_drop,ifelse(sum_comp[3] == 0,3,1))

      return(t_drop)

    }else if(length(treatment) == 2){

      flag <- TRUE

      # first examine the boundaries
      comp0 <- foreach (i0=1:N_grid_extra,.combine=c) %
        dopar% {
        com0 <- 0

        x_new <- matrix(Xgrid_extra[i0,],1,num_vars)
        pprr_x_grid_i0 <- SUBA_calc_pprr(x_new,
          SUBA_trained)
        pprr_x_grid_i0 <- c(pprr_x_grid_i0)
        pprr_x_grid_i0 <- pprr_x_grid_i0[-t_dropped]
        if (pprr_x_grid_i0[1] > pprr_x_grid_i0[2]) com0 <-
          1
        com0
      }

      sum_comp0 <- sum(comp0)

      if(sum_comp0 != 0 & sum_comp0 != 16) flag <- FALSE

      # Next,sequentially examine the grid values

      for(i1 in 1:round(N_grid / N_parallel)){

        if(flag){
          comp <- foreach (i2=1:N_parallel,.combine=c) %
```

```
            dopar% {
511           com <- 0
              i3 <- N_parallel * (i1-1)+i2

              x_new <- matrix(Xgrid[i3,],1,num_vars)
              pprr_x_grid_i3 <- SUBA_calc_pprr(x_new,
                 SUBA_trained)
516           pprr_x_grid_i3 <- c(pprr_x_grid_i3)
              pprr_x_grid_i3 <- pprr_x_grid_i3[-t_dropped]

              if (pprr_x_grid_i3[1] > pprr_x_grid_i3[2]) com
                  <- 1
              com
521         }
            sum_comp <- sum(comp)
            if (sum_comp/N_parallel != sum_comp0/
               N_grid_extra) flag <- FALSE
          } else break
        }
526     t_drop <- c()
        if (flag)
          t_drop <- ifelse(sum_comp == 0,1,2)
        return(t_drop)

531   }

  }


536 SUBA_train <- function(X,Y,Z,SUBA_trained,early_stop=
      FALSE){

    SUBA_trained_new <- SUBA_trained
    new_info <- SUBA_partition_likelihood(X,Y,Z,
       SUBA_trained)

541 SUBA_trained_new$part_lik <- new_info$part_lik
    SUBA_trained_new$subgroup_split_point <- new_info$
       subgroup_split_point
    SUBA_trained_new$subgroup_pprr <- new_info$
       subgroup_pprr
    SUBA_trained_new$n_obs <- length(Z)

546 if(early_stop){
      t_dropped <- SUBA_trained_new$t_dropped
      treatment <- SUBA_trained_new$treatment

      print(paste0("Currently remaining treatments: ",
         paste(treatment,collapse=" "),"."))
```

```
551
      t_drop <- SUBA_early_stop(SUBA_trained_new,
         N_parallel=200)
      if(length(t_drop) > 0){
        t_dropped <- c(t_dropped,treatment[t_drop])
        print(sprintf("After training,treatment %d is
           inferior than the other treatments and is
           dropped :-(..",treatment[t_drop]))
556     treatment <- treatment[-t_drop]
      }else{
        print("After training,no treatment is uniformly
           inferior than the other treatments. All
           treatment options could continue :-).")
      }
      SUBA_trained_new$t_dropped <- t_dropped
561   SUBA_trained_new$treatment <- treatment
    }

    return(SUBA_trained_new)
  }
566

  SUBA_prescribe <- function(x_new,SUBA_trained,early_stop
     =FALSE){

    N_new <- dim(x_new)[1]
571
    if(early_stop){
      treatment <- SUBA_trained$treatment
      t_dropped <- SUBA_trained$t_dropped
      if(length(treatment) == 1){
576     t_assigned <- rep(treatment,N_new)
      } else if(length(treatment) == 2){
        pprr_new <- SUBA_calc_pprr(x_new,SUBA_trained)
        pprr_new[,t_dropped] <- 0
        t_assigned <- apply(pprr_new,1,which.max)
581   }else if(length(treatment) == 3){
        pprr_new <- SUBA_calc_pprr(x_new,SUBA_trained)
        t_assigned <- apply(pprr_new,1,which.max)
      }
    }else{
586   pprr_new <- SUBA_calc_pprr(x_new,SUBA_trained)
      t_assigned <- apply(pprr_new,1,which.max)
    }
    return(t_assigned)
  }
591

  # SAMPLING SIMULATION ----
```

```
SUBA_sampling_sim <- function(max_num,init_num,wave_num,
   trial_name,num_vars,is_binary,num_treatm,data_file,
   early_stop=0,max_wave=10000,scaling="none",seed
   =29795,SUBA_trained=NULL,sample_index=NULL,wave_hist=
   NULL,interm=c(1000,2000,3000,4000),...){
  require(hopach)
  require(foreach)
  require(doParallel)
  require(doSNOW)

  # Starting point ----
  int_saved <- rep(0,length(interm))

  print(sprintf("Name of trial: %s.",trial_name))
  set.seed(seed)

  if(num_vars != length(is_binary)){
    stop("Warning: Number of variables does not match
       with length(is_binary) !")
  }

  if(num_treatm != 3 & early_stop){
    stop("Early stop can only handle num_treatm=3
       treatments")
  }

  # Reading data ----
  print("Reading data...")
  data <- read.csv(data_file,...)

  X_all <- as.matrix(data[,grepl("Variable",colnames(
     data))]) #rows are observed observations and
     columns are num_vars variables
  if(dim(X_all)[2] != num_vars){
    stop("Warning: Number of variables does not match
       with the specified num_vars!")
  }

  Y_all <- data$Outcome #outcomes of observed
     observations

  Z_all <- data$Treatment #treatments of observed
     observations
  if(max(Z_all) != num_treatm | length(table(Z_all)) !=
     num_treatm){
    stop("Warning: Number of treatments does not match
       with the specified num_treatm!")
  }
```

```
  print("Data loaded!")

# variables description ----
print(sprintf("Number of variables: %d.",num_vars))
for(k in 1:num_vars){
  if(is_binary[k]){
    print(sprintf("Variable %d is binary.",k))
  }else{
    print(sprintf("Variable %d is not binary.",k))
  }
}
print(sprintf("Number of treatments: %d.",num_treatm))

# SUBA EMPTY ----
if (is.null(SUBA_trained)){
  print("Building partitions...")
  list_part_prior <- SUBA_partition(num_vars,is_binary
     )
  print("Partitions built!")
  part <- list_part_prior$part
  part_prior <- list_part_prior$part_prior
  print("Building subgroups...")
  subgroup <- SUBA_subgroup(num_vars,is_binary)
  print("Subgroups built!")
  print("Assegning subgroups to partitions...")
  partsub <- SUBA_partsub(part,subgroup,num_vars)
  print("Subgroups assigned!")
  if(early_stop) Xgrid <- SUBA_Xgrid(num_vars,10)

  # Building empty model
  SUBA_trained <- list()
  SUBA_trained$num_vars <- num_vars
  SUBA_trained$is_binary <- is_binary
  SUBA_trained$num_treatm <- num_treatm
  SUBA_trained$part <- part
  SUBA_trained$part_prior <- part_prior
  SUBA_trained$subgroup <- subgroup
  SUBA_trained$partsub <- partsub
  if (early_stop) SUBA_trained$Xgrid <- Xgrid
  SUBA_trained$hp <- list(a=1,b=1)

  SUBA_trained$treatment <- 1:num_treatm
  SUBA_trained$t_dropped <- c()
  SUBA_trained$n_obs <- 0
  pretraining_obs <- NULL

  print("SUBA initial training finished.")
  saveRDS(SUBA_trained,file <- sprintf("
     SUBA_empty_model_%s.rds",trial_name))
```

```
    print(sprintf("Trained empty model saved in '
      SUBA_empty_model_%s.rds'.",trial_name))
  } else {
    pretraining_obs <- N_obs <- SUBA_trained$n_obs
    print(sprintf("SUBA empty model already trained.
      Number of observation: %d",SUBA_trained$n_obs))
  }

  # Scaling continuous variables and coding binary ones
    ----
  print("Scaling data...")
  if (scaling == "standard"){
    X_all[,!is_binary] <- scale(X_all[,!is_binary])
  } else if (scaling == "normal"){
    # Normalizing function ----
    normalized <- function(x,na.rm=TRUE) {
      (x-min(x,na.rm=na.rm))/(max(x,na.rm=na.rm)-min(x,
        na.rm=na.rm))
    }
    X_all[,!is_binary] <- apply(as.matrix(X_all[,!
      is_binary]),2,normalized)
  }
  print("Scaling completed!")

  print("Checking binary variables coding...")
  if(typeof(X_all[,as.logical(is_binary)]) == "character
    " ){
    for(i in (1:num_vars)[as.logical(is_binary)]){
      if(typeof(X_all[,i]) == "character"){
        X_all[,i] <- as.numeric(as.factor(X_all[,i]))-1
        print(sprintf("Variable %d successfully recoded!
          ",i))
      }
    }
    X_all <- as.matrix(apply(X_all,2,as.numeric))
  }
  print("Checking binary variables completed!")

  # SUBA TRAINING PRE-WAVES ----

  if(early_stop){
    if(num_treatm != 3){
      stop("Early stop can only handle num_treatm=3
        treatments")
    }
    ncores <- min(detectCores()-1,15)
    cl <- makeCluster(ncores)
    registerDoSNOW(cl)
  }
```

```
  if (is.null(sample_index)){
    sample_index <- sample(1:nrow(data),size=init_num) #
       initial training sample
721   N_obs <- init_num

    X <- X_all[sample_index,]
    Y <- Y_all[sample_index]
    Z <- Z_all[sample_index]
726
    SUBA_trained <- SUBA_train(X,Y,Z,SUBA_trained,
       early_stop)

    print(sprintf("Number of observations used for SUBA
       first training: %d.",N_obs))
    print("SUBA training finished. Initial round")
731   saveRDS(SUBA_trained,file=sprintf("
       SUBA_initial_trained_%s.rds",trial_name))
    print(sprintf("Initial trained model saved in '
       SUBA_initial_trained_%s.rds'.",trial_name))
  }

  # Wave building ----
736 nwave <- length(wave_hist)
  while ((N_obs<max_num) & (nwave<max_wave)) {
    nwave <- nwave+1
    print(paste0("Wave ",nwave))
    ind_pop <- setdiff(1:nrow(data),sample_index) #
       population from which extract wave sample
741
    # Wave observations
    ind_wave <- sample(ind_pop,size=wave_num); print(
       ind_wave)

    X_new <- X_all[ind_wave,]
746   Z_new <- Z_all[ind_wave]

    # Wave population
    ind_pop <- setdiff(ind_pop,ind_wave)
    X_pop <- X_all[ind_pop,]
751   Z_pop <- Z_all[ind_pop]

    # SUBA PRESCRIPTION ----
    print("Prescribing treatments...")
    t_assigned <- SUBA_prescribe(x_new=X_new,
       SUBA_trained=SUBA_trained,early_stop=early_stop)
756   print("Prescriptions completed!")

    # Observation substitution----
```

```
    print("Computing wrong observations...")
    to_be_subs <- (1:wave_num)[t_assigned!=Z_new] #wrong
        treatment observations
761 print("Wrong observations computed!")

    print("Substituting observations...")
    dist_substitutes <- NULL
    for (i in to_be_subs){
766   corr_treat <- Z_pop == t_assigned[i]
      index <- (1:length(Z_pop))[corr_treat]
      X_subs <- X_pop[index,]
      dist_vect <- distancevector(X=X_subs,y=as.numeric(
         X_new[i,]),d="euclid")
      dist_substitutes <- rbind(dist_substitutes,c(index
         [which.min(dist_vect)],min(dist_vect)))
771 }
    print("Observations substitution completed!")

    ind_wave_new <- unique(c(ind_wave[Z_new ==
       t_assigned],dist_substitutes[,1]))
    print(ind_wave_new)
776 wave_eff_size <- length(ind_wave_new)
    print(sprintf("Wave %d effective size: %d",nwave,
       wave_eff_size))
    wave_hist <- c(wave_hist,wave_eff_size)

    sample_index <- c(sample_index,ind_wave_new) #sample
        used in wave training
781
    # SUBA TRAINING ----
    if(early_stop){
      library(foreach)
      library(doParallel)
786   ncores <- min(detectCores()-1,15)
      cl <- makeCluster(ncores)
      registerDoSNOW(cl)
    }

791 N_obs <- length(sample_index)

    X <- X_all[sample_index,]
    Y <- Y_all[sample_index]
    Z <- Z_all[sample_index]
796
    SUBA_trained <- SUBA_train(X,Y,Z,SUBA_trained,
       early_stop)

    print(sprintf("Number of observations used for SUBA
       training: %d.",N_obs))
```

```
    print(sprintf("SUBA training finished. Wave %d",
       nwave))

      # Saving intermediate model ----
    for (i in 1:length(interm)) {
      if (int_saved[i] == 0 & interm[i]<N_obs){
        saveRDS(SUBA_trained,file=sprintf("SUBA_trained_
           %d_%d_%d.rds",init_num,wave_num,interm[i]))
        saveRDS(sample_index,file=sprintf("sample_index_
           %d_%d_%d.rds",init_num,wave_num,interm[i]))
        saveRDS(wave_hist,file=sprintf("wave_hist_%d_%d_
           %d.rds",init_num,wave_num,interm[i]))

        print(sprintf("Intemediate model with %d obs
           saved!",interm[i]))
        int_saved[i] <- 1
      }
    }
  }

  # Stop reason ----
  if (nwave == max_wave){
    print(sprintf("Stopped. Reached maximum waves number
       : %d waves",max_wave))
  } else if (N_obs>=max_num){
    print(sprintf("Stopped. Reached desired sample size
       number (%d): %d observations used",max_num,N_obs)
       )
  }

  # Output ----
  out <- list(sample=sample_index,model=SUBA_trained,
     wave_history=wave_hist)
  if (!is.null(pretraining_obs)) out$pretraining_obs <-
     pretraining_obs
  return(out)
}

# COMPARISON BETWEEN MODEL AND RANDOM SAMPLE ----
comparison<-function(data_file,mod_path,init=c
   (500,1000,1500),wave_size=c(50,100,150),N=seq
   (1000,6500,by=500),tests=TRUE,bilateral=FALSE,
   descriptive=TRUE,graphs="none",color="gray40"){
  data<-read.csv(data_file)

  # Initializing output ----
  out<-NULL
  if (descriptive | tests) name<-list(paste0("Init. size
      ",init),paste0("Wave size ",wave_size),paste0("
```

```
    Final size ",N)) else N<-N[length(N)]
  if (descriptive) out$nwaves<-out$sample_size<-array(
    dim=c(length(init),length(wave_size),length(N)),
    dimnames=name)
836 if (tests) out$prop_SUBA<-out$prop_rand_sample<-out$
    pvalue<-out$ztest<-array(dim=c(length(init),length(
    wave_size),length(N)), dimnames=name)

  for (n in N){
    set.seed(1234)
    if(graphs=="wave"){
841   par(mfrow=c(length(wave_size),length(init)))
    }

    print(paste0("n=",n))

846 for (i in init){
      print(paste0("i=",i))

      for (w in wave_size){
        print(paste0("w=",w))
851
        # Reading data files ----
        file_name<-paste0(mod_path,sprintf("
          sample_index_%d_%d_%d.rds",i,w,n))
        sample_index<-readRDS(file=file_name)[-1:i]
        file_name<-paste0(mod_path,sprintf("wave_hist_%
          d_%d_%d.rds",i,w,n))
856     wave_hist<-readRDS(file=file_name)

        # Selecting correct data indexes ----
        if (graphs=="wave") sample_graphs<-sample_index
        num<-length(sample_index)
861
        # Descriptive output ----
        if (descriptive){
          out$sample_size[which(i==init),which(w==
            wave_size),which(n==N)]<-num
          out$nwaves[which(i==init),which(w==wave_size),
            which(n==N)]<-length(wave_hist)
866     }

        # Test output ----
        if (tests){
          rand_sample<-sample(1:nrow(data),size = num,
            replace = F)
871
          out$prop_SUBA[which(i==init),which(w==
            wave_size),which(n==N)]<-p1<-mean(data$
```

```
            Outcome[sample_index])
          out$prop_rand_sample[which(i==init),which(w==
            wave_size),which(n==N)]<-p2<-mean(data$
            Outcome[rand_sample])

          out$ztest[which(i==init),which(w==wave_size),
            which(n==N)]<-zvalue<-(p1-p2)/sqrt((p1*(1
            -p1)+p2*(1-p2))/num)

          if (!bilateral) {
            out$pvalue[which(i==init),which(w==wave_size
              ),which(n==N)]<-pnorm(abs(zvalue),
              lower.tail=F)+0.5*I(zvalue<0)
          } else{
            out$pvalue[which(i==init),which(w==wave_size
              ),which(n==N)]<-pnorm(abs(zvalue),
              lower.tail=F)*2
          }
        }

        # Plot ----
        if (graphs=="wave"){
          cum_hist<-cumsum(wave_hist)
          if (n==N[length(N)]){
            SUBA_sample<-prop<-NULL

            for (wave in length(wave_hist):1){
              if (wave==1) starting<-0 else starting<-
                cum_hist[wave-1]

              index<-(starting+1):(starting+wave_hist[
                wave])
              SUBA_sample<-sample_graphs[index]
              prop<-c(prop,mean(data$Outcome[SUBA_sample
                ]))
            }

            # Single wave proportion ----
            if (graphs=="wave"){
              prop<-prop[length(prop):1]
              cummean_prop<- cumsum(prop*wave_hist)/
                cumsum(wave_hist)
              plot(y=prop,x=cum_hist,type = "b",col=
                color,
                  ylim=c(0.03,0.27),
                  ylab="Proporzione di successo",xlab="
                    Numero di osservazioni")
              lines(cbind(cum_hist,cummean_prop),lty=5,
                lwd=2)
```

```
906
                    #Parametri grafici
                    title(paste0(expression(omega),"=",w," ",
                       expression(n[init]),"=",i))
                    abline(h=mean(data$Outcome),lty=3,lwd=1)
                    leg<-c("Singola wave","Cumulata","
                       Popolazione")
911                 legend(legend=leg, inset=0.01, lty = c
                       (1,5,3), lwd=rep(1,2,1),
                           pch = c(1,NA,NA), "topright", bty =
                               "n")

                  }
                }
916           }
            }
          }
        }
      if (tests) out$differences<-out$prop_SUBA-out$
         prop_rand_sample
921
      if(!is.null(out)) return(out)
    }
```

Bibliografia

Azzalini, A. & Scarpa, B. (2012), *Data Analysis and Data Mining: An Introduction*, Oxford University Press, USA.

Hastie, T., Tibshirani, R. & Friedman, J. (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics, second edn, Springer New York.

Jech, T. (2003), *Set theory - The Third Millennium Edition, revised and expanded*, Springer.

Kuhn, M. (2018), *caret: Classification and Regression Training*. R package version 6.0-80.
URL: *https://CRAN.R-project.org/package=caret*

Liaw, A. & Wiener, M. (2002), 'Classification and regression by randomForest', *R News* **2**(3), 18–22.
URL: *https://CRAN.R-project.org/doc/Rnews/*

Microsoft Corporation & Weston, S. (2017*a*), *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.11.
URL: *https://CRAN.R-project.org/package=doParallel*

Microsoft Corporation & Weston, S. (2017*b*), *doSNOW: Foreach Parallel Adaptor for the 'snow' Package*. R package version 1.0.16.
URL: *https://CRAN.R-project.org/package=doSNOW*

Microsoft Corporation & Weston, S. (2017*c*), *foreach: Provides Foreach Looping Construct for R*. R package version 1.4.4.
URL: *https://CRAN.R-project.org/package=foreach*

Milborrow, S. (2018), *earth: Multivariate Adaptive Regression Splines*. R package version 4.6.3.
URL: *https://CRAN.R-project.org/package=earth*

Pace, L. & Salvan, A. (2001), *Introduzione alla statistica*, number v. 2 *in* 'Introduzione alla statistica', CEDAM.

R Core Team (2018), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *http://www.R-project.org/*

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C. & Müller, M. (2011), 'pROC: an open-source package for R and S+ to analyze and compare ROC curves', *BMC Bioinformatics* **12**, 77.

UCI Machine Learning Repository (2012), 'Bank marketing data set'.
URL: *https://archive.ics.uci.edu/ml/datasets/bank+marketing*

van der Laan, M. J. & Pollard, K. S. (2003), 'Hybrid clustering of gene expression data with visualization and the bootstrap', *Journal of Statistical Planning and Infererence* **117**, 275–303.

Venables, W. N. & Ripley, B. D. (2002), *Modern Applied Statistics with S*, fourth edn, Springer, New York.

Wickham, H. (2011), 'The split-apply-combine strategy for data analysis', *Journal of Statistical Software* **40**(1), 1–29.
URL: *http://www.jstatsoft.org/v40/i01/*

World Medical Association (2013), 'World medical association declaration of Helsinki: Ethical principles for medical research involving human subjects', *Journal of the American Medical Association* **310**(20), 2191–2194.

Xu, Y., Trippa, L., Muller, P. & Yuan, J. (2016), 'Subgroup-Based Adaptive (SUBA) designs for multi-arm biomarker trials', **8**(1), 159–180.

Ringraziamenti

Desidero ringraziare il relatore, prof. Bruno Scarpa, per avermi seguito nello sviluppo del metodo proposto e nella stesura del presente elaborato.

Inoltre ringrazio la prof.ssa Yanxun Xu, il prof. Yuan Ji e il dott. Tianjian Zhou per avermi fornito il materiale supplementare dell'articolo Xu et al. (2016) che mi ha aiutato nell'implementazione del metodo proposto in questa tesi.