Università degli studi di Padova
Dipartimento di Scienze Statistiche

Corso di Laurea Magistrale in
Scienze Statistiche


Tesi di Laurea

# Metodi di Classificazione Basati sulla Verifica d'ipotesi: Variazioni, Estensioni e Applicazioni

Relatore Prof. Livio Finos
Dipartimento di Psicologia dello Sviluppo e della Socializzazione

Laureando Giovanni Corradini
Matricola N° 1210593

Anno Accademico 2020/2021

# Indice

# Introduzione

La verifica d'ipotesi e la classificazione sono due discipline statistiche molto utilizzate e ben distinte fra loro: la prima viene adoperata in contesti dove si vuole valutare la veridicità di una ipotesi sulla base dei dati osservati, mentre la seconda in contesti nei quali si vuole prevedere, sulla base di informazioni note, l'esito di una variabile qualitativa (o classe) ignota. Sebbene queste due discipline vengano utilizzate in contesti differenti fra loro, hanno comunque molti strumenti in comune; infatti Liao e Akritas (2007) hanno sviluppato un metodo di classificazione basato sulla struttura della verifica d'ipotesi, metodo che rappresenta il fulcro degli argomenti di questa tesi. In particolare nel capitolo 1 verranno richiamati alcuni concetti di verifica d'ipotesi, di classificazione binaria e dell'LDA (*Linear Discriminant Analysis*) ed in seguito verrà proposto il metodo di classificazione tramite verifica d'ipotesi (*Test Based Classification*, o $TBC$) di Liao e Akritas, con successive analogie tra un caso particolare (semplificato) di questo metodo e l'LDA e con ulteriori chiarimenti, proposti per la prima volta in questa tesi, sui legami presenti tra la versione (non semplificata) della TBC e l'LDA. Nel capitolo 2 le prime tre sezioni sono dedicate alla presentazione di altrettante evoluzioni della TBC, mentre nella quarta viene proposto un metodo per estendere la TBC al caso con $p > n$ (chiamato *High Dimensional Test Based Classification*, abbreviato in $HDTBC$), che si basa su uno shrinkage della matrice di varianza e covarianza proposto da Schäfer e Strimmer (2005): l'obiettivo principale di questa tesi infatti è quello di proporre un metodo che

estenda la TBC al caso con $p > n$ (in particolare con $n$ piccolo) e che sia più efficace (in termini di capacità predittive) degli altri metodi *test-based*. Nel capitolo 3 vengono applicati vari metodi di classificazione (tra cui l'LDA e la TBC) su 20 dataset reali, di cui 10 con $n > p$ e 10 con $p > n$ (quest'ultimi appartenenti quasi esclusivamente al contesto del sequenziamento dei geni), sui quali vengono misurate e confrontate le performance predittive (in termini di accuratezza misurata tramite convalida incrociata) di tali modelli.

# Capitolo 1

# Teoria di Riferimento

## 1.1 Classificazione Binaria e Analisi Discriminante Lineare (LDA)

Sia $X^{tr} = (x_0, x_1)$ l'insieme dei dati di stima, dove con $x_0 = (x_{01}, x_{02}, ..., x_{0n_0})$ e $x_1 = (x_{11}, x_{12}, ..., x_{1n_1})$ si indicano due campioni casuali generati da $X_0$ e da $X_1$ rispettivamente, e sia $X^{ts}$ una nuova osservazione generata o da $X_0$ o da $X_1$. L'obiettivo della classificazione binaria è quello di assegare $X^{ts}$ ad una delle due popolazioni (o classi) relative a $X_0$ e a $X_1$ (indicate con 0 e 1), utilizzando le informazioni contenute in $X^{tr}$.

L'analisi discriminante lineare, nota come LDA, è il metodo di classificazione più antico e fu inventato da Fisher nel 1936. Nell'LDA si assume che $X_0 \sim N_p(\mu_0, \Sigma)$ e che $X_1 \sim N_p(\mu_1, \Sigma)$, con $p \in [1, n_0 + n_1)$ e con $\Sigma$ invertibile. Indicando con $\pi_0$ e $\pi_1$ le probabilità a priori per le due classi (con $\pi_0 + \pi_1 = 1$), la regola definita dall'LDA assegna $X^{ts}$ alla classe 0 se:

$$(\mu_0 - \mu_1)' \Sigma^{-1} X^{ts} - \frac{1}{2} (\mu_0 - \mu_1)' \Sigma^{-1} (\mu_0 + \mu_1) > \ln \left( \frac{\pi_1}{\pi_0} \right), \qquad (1.1)$$

altrimenti assegna $X^{ts}$ alla classe 1; dove il membro di sinistra è il log rap-

porto fra le verosimiglianze di $X^{ts}$ nelle due classi. Sostituendo a $\mu_0$ e a $\mu_1$ le medie (o vettori di medie) campionarie di $X_0$ e di $X_1$ ($\bar{X}_0$ e $\bar{X}_1$) e a $\Sigma$ la varianza (o matrice di varianza e covarianza) pooled $S = \frac{(n_0-1)S_0+(n_1-1)S_1}{n_0+n_1-2}$, dove $S_0$ e $S_1$ sono le varianze (o matrici di varianze e covarianze) campionarie di $X_0$ e $X_1$, la Regola 1.1 si ridefinisce di conseguenza, assegnando $X^{ts}$ alla classe 0 se:

$$\left(\bar{X}_0 - \bar{X}_1\right)' S^{-1} X^{ts} - \frac{1}{2}\left(\bar{X}_0 - \bar{X}_1\right)' S^{-1}\left(\bar{X}_0 + \bar{X}_1\right) > \ln\left(\frac{n_1}{n_0}\right). \qquad (1.2)$$

## 1.2 Instance Based Learning

Oltre alla classica suddivisione degli algoritmi di apprendimento statistico in *supervisionato* e *non supervisionato* e alla un po' meno comune distinzione di tali algoritmi in *batch* (tutti i dati sono disponibili) e *online* (i modelli imparano in maniera incrementale da un flusso di dati), esiste anche la separazione in *instance-based* e *model-based*. I metodi di apprendimento *instance-based* (noti anche come *memory-based*), a differenza dei metodi *model-based*, invece che creare una regola di classificazione utilizzando i dati dell'insieme di stima, semplicemente salvano in memoria tali dati. In seguito, nel momento della previsione, invece che esplicitare la stessa regola di classificazione (stimata utilizzando appunto i dati dell'insieme di stima) su tutte le osservazioni dell'insieme di verifica, viene creata, sempre partendo dall'insieme di stima, una regola specifica per ogni osservazione dell'insieme di verifica, basandosi generalmente su distanze o semplici regole. Il metodo di apprendimento *instance-based* più famoso è il *k-nearest neighbor*, che nel contesto di classificazione binaria e nella sua versione più semplice consiste nel trovare le $k$ osservazioni più vicine (e.g. con minor distanza euclidea) a $X^{ts}$ e, se almeno $\frac{k}{2}$ *nearest neighbor* appartengono a $x_0$ allora si assegna $X^{ts}$ alla classe 0, in caso contrario si assegna $X^{ts}$ alla classe 1. La maggior parte dei modelli più popolari invece (come l'LDA, i modelli lineari generalizzati, la random forest...) sono metodi di apprendimento *model-based*.

## 1.3 Test Based Classification (TBC)

### 1.3.1 Richiami di Verifica d'Ipotesi

Si considerino due campioni casuali semplici indipendenti $x_0 = (x_{01}, x_{02}, ..., x_{0n_0})$ e $x_1 = (x_{11}, x_{12}, ..., x_{1n_1})$, generati da due variabili casuali $X_0 \sim N(\mu_0, \sigma^2)$ e $X_1 \sim N(\mu_1, \sigma^2)$ rispettivamente. Si desidera verificare $H_0 : \mu_0 = \mu_1$ contro l'alternativa che le medie siano diverse. Indicando con $\bar{X}_0$ e $\bar{X}_1$ e con $S_0$ e $S_1$ la media e la varianza campionarie di $x_0$ e di $x_1$ rispettivamente e con $S = \frac{(n_0-1)S_0+(n_1-1)S_1}{n_0+n_1-2}$ la varianza pooled, si ha che $t = \frac{(\bar{X}_0-\bar{X}_1)}{\sqrt{S(\frac{1}{n_0}+\frac{1}{n_1})}} \overset{H_0}{\sim} t_{n_0+n_1-2}$, ovvero, sotto $H_0$, $t$ si distribuisce come una t di Student con $(n_0+n_1-2)$ gradi di libertà. Per decidere se l'ipotesi può venire rifiutata o meno, dopo aver osservato una realizzazione di $x_0$ e di $x_1$, si sostituiscono le quantità teoriche $\bar{X}_0$, $\bar{X}_1$ e $S$ con le loro corrispettive quantità empiriche, ottenendo un valore relativo ad una realizzazione della statistica $t$, chiamato $t^{oss}$. Fissato un livello di significatività $\alpha$ e indicando con $F_t(.)$ la funzione di ripartizione della $t$ di *Student* (con $(n_0 + n_1 - 2)$ *g.d.l.*) e con $\alpha^{oss}$ il livello di significatività osservato, si tende a rifiutare l'ipotesi nulla se $\alpha^{oss} = 2\ min\ \{F_t(t^{oss}),\ (1 - F_t(t^{oss}))\} < \alpha$.

Considerando la generalizzazione multivariata dell'esempio precedente, ora si assume che $X_0 \sim N_p(\mu_0, \Sigma)$ e che $X_1 \sim N_p(\mu_1, \Sigma)$, con $1 < p < n_0 + n_1$. In questo caso si desidera verificare $H_0 : \mu_{0j} = \mu_{1j}\ \ per\ j = 1, ..., p$, contro l'alternativa che almeno una coppia di medie sia diversa. Indicando come prima con $\bar{X}_0$ e $\bar{X}_1$ e con $S_0$ e $S_1$ il vettore delle medie e la matrice di varianza e covarianza campionarie di $x_0$ e $x_1$ rispettivamente e con $S = \frac{(n_0-1)S_0+(n_1-1)S_1}{n_0+n_1-2}$ la matrice di varianza e covarianza pooled, l'estensione multivariata della t di Student è $T = \frac{n_0 n_1}{n_0+n_1}\left(\bar{X}_0 - \bar{X}_1\right)' S^{-1}\left(\bar{X}_0 - \bar{X}_1\right) \overset{H_0}{\sim} T^2_{(p,\ n_0+n_1-2)}$, ovvero, sotto $H_0$, $T$ si distribuisce come una $T^2$ di Hotelling con $p$ e $(n_0 + n_1 - 2)$ gradi di libertà. Anche qui, come nel caso precedente,

dopo aver sostituito le quantità teoriche con quelle empiriche ed aver ottenuto una realizzazione della $T^2$ di Hotelling ($T^{oss}$), indicando con $\alpha$ il livello di significatività fissato e con $F_T$ la funzione di ripartizione della $T^2$ di Hotelling, si propende per il rifiuto dell'ipotesi di uguaglianza delle medie se $\alpha^{oss} = (1 - F_T(T^{oss})) < \alpha$.

### 1.3.2 Test Based Classification (TBC)

Liao e Akritas (2007) hanno proposto un metodo di classificazione basato sui test d'ipotesi, chiamato appunto *test based classification (TBC)*. Indicando, come in precedenza, con $X^{tr} = (x_0, x_1)$ l'insieme dei dati di stima (con $x_0$ e $x_1$ generati da $X_0$ e $X_1$ rispettivamente) e con $X^{ts}$ una nuova osservazione, l'algoritmo della TBC consiste nei seguenti passi:

1. Si effettua un test tra $(x_0, X^{ts})$ e $x_1$ per la verifica d'ipotesi di uguaglianza delle medie dei due campioni, ottenendo un livello di significatività osservato, indicato con $\alpha_0^{oss}$;
2. Si effettua un test tra $x_0$ e $(X^{ts}, x_1)$ per la verifica d'ipotesi di uguaglianza delle medie dei due campioni, ottenendo un livello di significatività osservato, indicato con $\alpha_1^{oss}$;
3. Se $\alpha_0^{oss} < \alpha_1^{oss}$ allora si assegna $X^{ts}$ alla classe 0,
   altrimenti si assegna $X^{ts}$ alla classe 1.

L'algoritmo precedente può essere utilizzato sia con campioni univariati che multivariati e, come ipotesi da verificare nei primi due passi dell'algoritmo, oltre all'uguaglianaza delle medie possono essere verificate altre uguaglianze tra i due campioni (come l'uguaglianza fra le mediane o fra le distribuzioni), con conseguente scelta di un test adatto all'ipotesi da verificare (anche test non parametrici, test che sfruttano metodi di ricampionamento per il calcolo di $\alpha^{oss}$ ...), rendendo quindi il metodo TBC molto generale e variegato.

L'intuizione dietro alla TBC risiede nell'idea che il test dovrebbe fornire un livello di significatività osservato inferiore quando si alloca $X^{ts}$ nella classe giusta rispetto a quando la si alloca nella classe sbagliata. La precedente intuizione, nel caso in cui si utilizzino il t-test o la $T^2$ di Hotelling, è motivata dalla presenza di due forze sinergiche relative all'errata classificazione della nuova osservazione:

1. Quando si alloca $X^{ts}$ nella classe giusta la differenza delle medie fra i due gruppi dovrebbe essere superiore rispetto a quando la si alloca nella classe sbagliata;

2. Quando si alloca $X^{ts}$ nella classe sbagliata la varianza campionaria all'interno di questa classe dovrebbe aumentare maggiormente rispetto a quanto dovrebbe aumentare quando si alloca $X^{ts}$ nella classe giusta.

La TBC è strettamente connessa con l'LDA; infatti Liao e Akritas (2007) hanno mostrato che, in un contesto univariato con classi bilanciate ($n_0 = n_1 = n$), una versione semplificata della TBC che utilizza il t-test (pooled) equivale all'LDA. Infatti, utilizzando nei primi due passi dell'algoritmo della TBC un t-test e assumendo quindi che $x_0$ e $x_1$ siano generati da $X_0 \sim N(\mu_0, \sigma^2)$ e da $X_1 \sim N(\mu_1, \sigma^2)$ rispettivamente, con la varianza dei due campioni calcolata utilizzando solamente i dati dell'insieme di stima (e quindi senza $X^{ts}$), e indicando con

$t_0 = \frac{\frac{n\bar{X}_0 + X^{ts}}{n+1} - \bar{X}_1}{\sqrt{S\left(\frac{1}{n+1} + \frac{1}{n}\right)}}$ e $t_1 = \frac{\bar{X}_0 - \frac{n\bar{X}_1 + X^{ts}}{n+1}}{\sqrt{S\left(\frac{1}{n} + \frac{1}{n+1}\right)}}$ i t-test semplificati dei primi due

passi dell'algoritmo (dove $\bar{X}_0$, $\bar{X}_1$ e $S$ sono le usuali medie e varianza pooled campionarie), la regola di classificazione della TBC risulta essere:

$$\begin{aligned} &\alpha_0^{oss} < \alpha_1^{oss} \Leftrightarrow |t_0| > |t_1| \Leftrightarrow \left|\frac{n\bar{X}_0 + X^{ts}}{n+1} - \bar{X}_1\right| > \left|\bar{X}_0 - \frac{n\bar{X}_1 + X^{ts}}{n+1}\right| \\ &\Leftrightarrow \left|n\left(\bar{X}_0 - \bar{X}_1\right) + \left(X^{ts} - \bar{X}_1\right)\right|^2 > \left|n\left(\bar{X}_0 - \bar{X}_1\right) + \left(\bar{X}_0 - X^{ts}\right)\right|^2. \end{aligned} \tag{1.3}$$

L'ultima disequazione dell'Espressione 1.3 coincide (dopo qualche elaborazione algebrica) con la Disequzione 1.2, mostrando quindi l'uguaglianza tra la versione semplificata della TBC (classi bilanciate e t-test con varianza pooled calcolata solo con l'insieme di stima) e l'LDA. Questa equivalenza è valida anche in un contesto multivariato (anche qui come nel caso univariato l'equivalenza è valida solo se le classi sono bilanciate), utilizzando una versione semplificata della TBC che si basa sulla $T^2$ di Hotelling e assumendo quindi che $x_0$ e $x_1$ siano generati da $X_0 \sim N_p(\mu_0, \Sigma)$ e da $X_1 \sim N_p(\mu_1, \Sigma)$ rispettivamente, anche qui calcolando la varianza pooled utilizzando solamente i dati nell'insieme di stima. Infatti, indicando con

$$T_0 = \left[\frac{n\bar{X}_0 + X^{ts}}{n+1} - \bar{X}_1\right]' \left[\left(\frac{1}{n+1} + \frac{1}{n}\right) S\right]^{-1} \left[\frac{n\bar{X}_0 + X^{ts}}{n+1} - \bar{X}_1\right] \quad \text{e}$$

$$T_1 = \left[\bar{X}_0 - \frac{n\bar{X}_1 + X^{ts}}{n+1}\right]' \left[\left(\frac{1}{n} + \frac{1}{n+1}\right) S\right]^{-1} \left[\bar{X}_0 - \frac{n\bar{X}_1 + X^{ts}}{n+1}\right]$$

le $T^2$ di Hotelling semplificate relative ai primi due passi dell'algoritmo della TBC, la regola di classificazione della TBC risulta essere:

$$\begin{aligned} \alpha_0^{oss} < \alpha_1^{oss} &\Leftrightarrow T_0 > T_1 \\ &\Leftrightarrow [n(\bar{X}_0 - \bar{X}_1) + (X^{ts} - \bar{X}_1)]' \; S^{-1} \; [n(\bar{X}_0 - \bar{X}_1) + (X^{ts} - \bar{X}_1)] \\ &> [n(\bar{X}_0 - \bar{X}_1) + (\bar{X}_0 - X^{ts})]' \; S^{-1} \; [n(\bar{X}_0 - \bar{X}_1) + (\bar{X}_0 - X^{ts})]; \end{aligned} \tag{1.4}$$

dove anche in questo caso con $\bar{X}_0$, $\bar{X}_1$ e $S$ si indicano i vettori delle medie campionarie e la matrice di varianza e covarianza pooled. Anche l'ultima disequazione dell'Espressione 1.4 coincide con la Disequazione 1.2 e pertanto, anche nel contesto multivariato, la TBC semplificata, in caso di classi bilanciate e utilizzando la $T^2$ di Hotelling, coincide con l'LDA.

Come detto in precedenza, sempre assumendo che $x_0$ e $x_1$ siano generati da $X_0 \sim N_p(\mu_0, \Sigma)$ e da $X_1 \sim N_p(\mu_1, \Sigma)$, Liao e Akritas (2007) hanno mostrato che l'uguaglianza fra la TBC basata sul t-test (sulla $T^2$ di Hotelling) e l'LDA non rimane valida in caso di classi non perfettamente bilanciate. Inoltre, se nella TBC non viene utilizzata la versione semplificata

per la varianza (utilizzando solo i dati dell'insieme di stima), la sua regola di classificazione non è più basata (come nell'LDA) sull'allocare $X^{ts}$ nella classe dove la probabilità a posteriori (o verosimiglianza in caso di classi bilanciate) è più elevata, ma consiste invece nell'effettuare due rapporti fra le verosimiglianze delle due classi (un rapporto con $X^{ts}$ in una classe e uno con $X^{ts}$ nell'altra) ed allocare $X^{ts}$ nella classe dove il suo inserimento ha generato il rapporto fra le verosimiglianze più elevato (i.e. $\alpha^{oss}$ più piccolo). Infatti, indicando con $S_0^{ts}$ e $S_1^{ts}$ le varianze campionarie di $(x_0, X^{ts})$ e di $(X^{ts}, x_1)$ rispettivamente, con $S^0 = \frac{n_0 S_0^{ts} + (n_1-1)S_1}{n_0+n_1-1}$ e con $S^1 = \frac{(n_0-1)S_0 + n_1 S_1^{ts}}{n_0+n_1-1}$ le varianze pooled (ottenute come medie pesate fra $S_0^{ts}$ e $S_1$ e fra $S_0$ e $S_1^{ts}$) e con

$$T_0 = \left[\frac{n_0\bar{X}_0+X^{ts}}{n_0+1} - \bar{X}_1\right]' \left[\left(\frac{1}{n_0+1} + \frac{1}{n_1}\right) S^0\right]^{-1} \left[\frac{n_0\bar{X}_0+X^{ts}}{n_0+1} - \bar{X}_1\right] \quad \text{e}$$

$$T_1 = \left[\bar{X}_0 - \frac{n_1\bar{X}_1+X^{ts}}{n_1+1}\right]' \left[\left(\frac{1}{n_0} + \frac{1}{n_1+1}\right) S^1\right]^{-1} \left[\bar{X}_0 - \frac{n_1\bar{X}_1+X^{ts}}{n_1+1}\right],$$

l'Espressione 1.4 complicata (senza semplificazioni) diventa:

$$\begin{aligned} &\alpha_0^{oss} < \alpha_1^{oss} \Leftrightarrow T_0 > T_1 \\ &\Leftrightarrow \left[\frac{n_0\bar{X}_0 + X^{ts}}{n_0+1} - \bar{X}_1\right]' \left[\left(\frac{1}{n_0+1} + \frac{1}{n_1}\right) S^0\right]^{-1} \left[\frac{n_0\bar{X}_0 + X^{ts}}{n_0+1} - \bar{X}_1\right] \\ &> \left[\bar{X}_0 - \frac{n_1\bar{X}_1 + X^{ts}}{n_1+1}\right]' \left[\left(\frac{1}{n_0} + \frac{1}{n_1+1}\right) S^1\right]^{-1} \left[\bar{X}_0 - \frac{n_1\bar{X}_1 + X^{ts}}{n_1+1}\right]. \end{aligned} \tag{1.5}$$

Quindi la TBC può anche essere vista come una sequenza di coppie di LDA (una coppia per ogni osservazione dell'insieme di verifica), dove si alloca $X^{ts}$ nella classe relativa all'LDA (fra le due) che ha generato l'iperpiano che separa meglio le due classi, che è quello relativo al valore della $T^2$ di Hotelling (del rapporto di verosimiglianza) più elevato e al livello di significatività osservato più piccolo. Pertanto la differenza sostanziale tra la TBC (basata sul t-test o sulla $T^2$ di Hotelling) e l'LDA è che nell'LDA viene stimata una sola regola di classificazione (utilizzando i dati dell'insieme di stima) che viene poi generalizzata su tutte le unità dell'insieme di verifica (model-based), mentre nella TBC per ogni osservazione dell'insieme di verifica viene stimata

una regola di classificazione diversa (instance-based). Questa differenza fra i due approcci può essere un vantaggio per la TBC in termini di efficacia del modello, specialmente in presenza di poche osservazioni, in quanto a differenza dell'LDA viene tenuto in considerazione anche l'effetto che può avere l'errata assegnazione di una nuova osservazione sulla varianza fra le due classi, tuttavia è sicuramente uno svantaggio in termini di efficienza computazionale del modello, in quanto il tempo che occorre alla TBC per effettuare una previsione su un insieme di $k$ osservazioni è approssivamente $2k$ volte il tempo che impiega l'LDA per fare una previsione sugli stessi dati.

# Capitolo 2

# Evoluzioni della Test Based Classification

## 2.1 Test Based Classification per la Segmentazione di Immagini

Ghimire e Wang (2012) hanno sviluppato il metodo di Liao e Akritas (2007) specificatamente per la segmentazione di immagini, ovvero per la classificazione di determinate regioni di una o più immagini utilizzando la gradazione di grigio (variabile unidimensionale) dei pixel (unità statistiche). Infatti, in tale contesto, vi sono molti pixel per cui l'appartenenza ad una determinata regione è ben determinata (i.e. classi ben separate) e, per questi pixel, entrambi gli $\alpha^{oss}$ dei primi due passi dell'algoritmo della TBC risultano molto vicini a zero, diminuendo pertanto la capacità predittiva del modello. Per cercare di risolvere questo problema Ghimire e Wang (2012) hanno modificato il terzo passo dell'algoritmo TBC come segue:

3.1 Se $max(\alpha_0^{oss}, \alpha_1^{oss}) > \epsilon$ allora:

3.1.1 Se $\alpha_0^{oss} > \alpha_1^{oss}$ allora assegno $X^{ts}$ alla classe 0;

3.1.2 Altrimenti assegno $X^{ts}$ alla classe 1;

3.2 Se invece $max(\alpha_0^{oss}, \alpha_1^{oss}) \leq \epsilon$ allora:

3.2.1 Se $|X^{ts} - \bar{X}_0| < |X^{ts} - \bar{X}_1|$ allora assegno $X^{ts}$ alla classe 0;

3.2.2 Altrimenti assegno $X^{ts}$ alla classe 1;

dove $\bar{X}_0$ e $\bar{X}_1$ sono le usuali medie campionarie relative a $X_0$ e a $X_1$, $X^{ts}$ è la nuova osservazione la cui classe di appartenenza è ignota, $\alpha_0^{oss}$ e $\alpha_1^{oss}$ sono i livelli di significatività osservati ottenuti nei primi due passi dell'algoritmo della TBC utilizzando il t-test, $\epsilon$ è una certa soglia fissata (e.g. $\epsilon = 0.001$) e $|X^{ts} - \bar{X}_0|$ e $|X^{ts} - \bar{X}_1|$ sono semplicemente i valori assoluti della differenza fra i due valori, in quanto tale metodo è stato applicato solamente in contesti univariati.

## 2.2 Interpoint Distance Classification (IDC)

Guo e Modarres (2019) hanno adattato il metodo di Liao e Akritas (2007) per poterlo applicare in contesti dove la dimensionalità della matrice del disegno è superiore alla sua dimensione (i.e $p > n$), in particolare si sono concentrati su insiemi di dati ad elevata dimensionalità e con tutte le esplicative discrete ordinali. Pertanto è stata proposta una versione alternativa alla TBC (ma che ne preserva la struttura di fondo), chiamata *IDC* (*Interpoint Distance Classification*), che sfrutta appunto le *Interpoint Distances* o *IPDs*. Indicando come in precedenza con $x_0$ e $x_1$ due campioni casuali (di dimensione $n_0 \times p$ e $n_1 \times p$) da $X_0 \sim F_0$ e da $X_1 \sim F_1$ e con $||x_0|| = (x_0' \; x_0)^{\frac{1}{2}}$ e $||x_1|| = (x_1' \; x_1)^{\frac{1}{2}}$ la norma euclidea di $x_0$ e di $x_1$, la IPD tra l'i-esimo e il j-esimo elemento di $x_0$ risulta essere $d_{x_0}^{ij} = ||x_{0i} - x_{0j}||$. Inoltre, indicando con $d_{x_1}^{ij} = ||x_{1i} - x_{1j}||$ la IPD tra l'i-esimo e il j-esimo elemento di $x_1$ e con $d_{(x_0,x_1)}^{ij} = ||x_{0i} - x_{1j}||$ la

IPD tra l'i-esimo di $x_0$ e il j-esimo elemento di $x_1$, le IPD medie entro i due campioni $x_0$ e $x_1$ risultano essere $\bar{d}_{(x_0)} = \binom{n_0}{2}^{-1} \sum_{i=1}^{(n_0-1)} \sum_{j=i+1}^{n_0} d_{x_0}^{ij}$ e $\bar{d}_{(x_1)} = \binom{n_1}{2}^{-1} \sum_{i=1}^{(n_1-1)} \sum_{j=i+1}^{n_1} d_{x_1}^{ij}$ rispettivamente, mentre la IPD media fra $x_0$ e $x_1$ è pari a $\bar{d}_{(x_0,x_1)} = (n_0 n_1)^{-1} \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} d_{(x_0,x_1)}^{ij}$.Con queste premesse Baringhaus e Franz (2004), per verificare $H_0 : F_0 = F_1$ contro l'alternativa che le due distribuzioni siano diverse, hanno proposto una statistica che si basa sulle IPD, chiamata appunto statistica di Baringhaus e Franz:
$BF(x_0, x_1) = \frac{n_0 n_1}{n_0+n_1}[\bar{d}_{(x_0,x_1)} - 2\binom{n_0}{2} n_0^{-2} \bar{d}_{(x_0)} - 2\binom{n_1}{2} n_1^{-2} \bar{d}_{(x_1)}]$.
La distribuzione della statistica di Baringhaus e Franz viene calcolata tramite metodi di ricampionamento e la regola di tale test è che si rifiuta l'ipotesi nulla per valori grandi della statistica. Dopo tali premesse e indicando, come di consuetudine, con $X^{ts}$ la nuova osservazione da classificare, l'algoritmo della IDC diventa:

1. Calcolare $T = BF(x_0, x_1)$;
2. Calcolare $T_0 = BF((x_0, X^{ts}), x_1)$;
3. Calcolare $T_1 = BF(x_0, (X^{ts}, x_1))$;
4. Se $|\frac{T_0-T}{T_1-T}| < 1$ allora si assegna $X^{ts}$ alla classe 0,
   altrimenti si assegna $X^{ts}$ alla classe 1.

La differenza principale della IDC rispetto alla TBC risiede nell'utilizzo delle IPD invece che dei dati originali, fatto che permette di lavorare anche in contesti con $p > n$ grazie al collasso, causato appunto dall'utilizzo delle IPD, di tutte e $p$ le dimensioni dei dati su di una soltanto. L'ulteriore differenza risiede nell'utilizzo di $|\frac{T_0-T}{T_1-T}|$, invece che confrontare direttamente $\alpha_0^{oss}$ e $\alpha_1^{oss}$: questa scelta è motivata dal fatto che, come era già emerso nello studio di Ghimire e Wang (2012), mediante il confronto diretto dei livelli di sgnificatività osservati non vi è un'adeguata distinzione fra le due classi (soprattutto se queste sono ben separate), mentre utilizzando $|\frac{T_0-T}{T_1-T}|$ questo problema viene superato.

## 2.3 Instance Based Classification (IBC)

He et al. (2019) hanno sviluppato il metodo di Guo e Modarres (2019) per tenere in considerazione che i metodi basati sulla TBC sono una classe di modelli di apprendimento instance-based e infatti hanno chiamato il loro modello di classificazione *Instance Based Classifier Through Hypothesis Testing* (*IBC*). In particolare He et al. (2019) sostengono che la presenza di outliers (in particolare di osservazioni distanti da entrambe le classi) e di osservazioni irrilevanti (molte osservazioni ripetute o con informazioni ridondanti) può indebolire le performance di un modello instance-based. Pertanto, per alleviare questo problema, He et al. (2019) propongono un passo antecedente a tutti i passi dell'algoritmo IDC che consiste nel calcolare i $k$ nearest neighbor di $X^{ts}$ in $x_0$, i $k$ nearest neighbor di $X^{ts}$ in $x_1$ e di sostituire questi $2k$ nearest neighbor con l'intero insieme di stima (chiaramente questo passo va ripetuto per ogni osservazione dell'insieme di verifica).

Un'ulteriore modifica che He et al. (2019) hanno apportato al metodo IDC consiste nella statistica utilizzata per verificare $H_0 : F_0 = F_1$. In particolare, indicando ora con $x_0^k$ e $x_1^k$ i $k$ nearest neighbor di $X^{ts}$ in $x_0$ e $x_1$, con $d_{(x_0^k, X^{ts})}$ e $d_{(x_1^k, X^{ts})}$ i vettori k-dimensionali delle IPD tra $X^{ts}$ e ogni elemento di $x_0^k$ e di $x_1^k$ rispettivamente e con $F(d_{(x_0^k, X^{ts})})$ e $F(d_{(x_1^k, X^{ts})})$ le distribuzioni di tali IPD, l'algoritmo della IBC diventa:

1. Testare $H_0 : F(d_{(x_0^k, X^{ts})}) = F(d_{(X^{ts}, x_1^k)})$ contro l'alternativa $H_1^0 : F(d_{(x_0^k, X^{ts})}) < F(d_{(X^{ts}, x_1^k)})$ utilizzando il test dei ranghi con segno di Wilcoxon e ottenendo $\alpha_0^{oss}$;

2. Testare $H_0 : F(d_{(x_0^k, X^{ts})}) = F(d_{(X^{ts}, x_1^k)})$ contro l'alternativa $H_1^1 : F(d_{(x_0^k, X^{ts})}) > F(d_{(X^{ts}, x_1^k)})$ utilizzando il test dei ranghi con segno di Wilcoxon e ottenendo $\alpha_1^{oss}$;

3. Se $\alpha_0^{oss} < \alpha_1^{oss}$ allora si assegna $X^{ts}$ alla classe 0,
   altrimenti si assegna $X^{ts}$ alla classe 1.

Infine, a differenza dell'articolo di Liao e Akritas (2007) e dei suoi successori, He et al. (2019) non hanno misurato le performance del loro modello (IBC) utilizzando dati simulati o in contesti specifici (come la segmentazione di immagini o l'analisi di osservazioni discrete multidimensionali), ma hanno invece deciso di misurarle in situazioni reali su dati derivati da ambiti diversi. Infatti hanno valutato le capacità predittive della IBC su 40 dataset ottenuti dalle repository di dati quali *UCI Machine Learning repository* di Asuncion e Newman (2007) e *Keel* di Alcalá-Fdez et al. (2011); tuttavia, anche se le IPD sono state introdotte per poter applicare il modello su dataset con più variabili esplicative che osservazioni, neanche uno dei 40 dataset utilizzati era di questo tipo.

## 2.4 Metodo Proposto: High Dimensional Test Based Classification (HDTBC)

In questa tesi viene proposto un ulteriore metodo di classificazione binaria basato sulla TBC (chiamato *High Dimensional Test Based Classification* abbreviato in *HDTBC*), che funziona anche con $p > n$ ma senza ricorrere a tecniche come le IPD, che non tengono in considerazione la correlazione fra i predittori. Il metodo è un'estensione diretta della TBC basata sulla $T^2$ di Hotelling, sviluppata specificatamente per contesti con poche osservazioni e molti predittori (e.g. sequenziamento dei geni), che consiste nell'operare uno *shrinkage* della matrice di varianza e covarianza (della $T^2$ di Hotelling) che ne garantisce l'invertibilità anche con $p > n$. Il metodo di shrinkage utilizzato è quello proposto da Schäfer e Strimmer (2005) ed è stato sviluppato appunto per contesti genomici dove la dimensionalità dei dati (i geni) è di molte volte superiore al numero di osservazioni; tale metodo

sfrutta i risultati di Ledoit e Wolf (2003), che introdussero uno stimatore della matrice di varianza e covarianza che consiste in una media pesata (con il peso rappresentato dal parametro di shrinkage) tra la matrice di varianza e covarianza campionaria e la matrice identica, dove il parametro di shrinkage viene calcolato analiticamente minimizzando l'errore quadratico medio di tale stimatore. Più precisamente in questa tesi vengono utilizzati due stimatori diversi per la matrice di varianza e covarianza, uno per gli elementi dentro alla diagonale e uno per quelli fuori, ma entrambi basati sul metodo di shrinkage di Ledoit e Wolf (2003): per gli elementi fuori dalla diagonale (le covarianze) viene effettuato lo shrinkage proposto appunto da Schäfer e Strimmer (2005), mentre per gli elementi sulla diagonale (le varianze) viene proposta una rivisitazione più robusta di tale metodo, sviluppata sempre da Opgen-Rhein e Strimmer (2007).

Nello specifico, indicando con $X^{tr}$ l'insieme dei predittori dei dati di stima (con $n$ righe e $p$ colonne), con $\sigma^2 = (\sigma_1^2, ..., \sigma_p^2)$ le varianze dei singoli predittori, con $\sigma_m^2$ la mediana di $\sigma^2$, con $s$ uno stimatore di $\sigma^2$ e con $s_m$ la mediana di $s$, lo stimatore della $j$-esima componente di $\sigma^2$ dato dallo shrinking tra $s_j$ e $s_m$ proposto da Opgen-Rhein e Strimmer (2007) è: $s_j^\star = \lambda_v s_m + (1 - \lambda_v)s_j$, con $\lambda_v \in [0, 1]$. Assumendo l'esistenza e la finitezza dei primi due momenti di $s$ e di $s_{med}$, utilizzando il metodo di Ledoit e Wolf (2003) si ottiene il $\lambda_v$ che minimizza l'errore quadratico medio di $s^\star$, che è: $EQM(s^\star) = E[\sum_{j=1}^p (s_j^\star - \sigma_j^2)^2] = \sum_{j=1}^p [Var(s_j^\star) + (E(s_j^\star) - \sigma_j^2)^2] = \sum_{j=1}^p [Var(\lambda_v s_m + (1-\lambda_v)s_j) + (E(\lambda_v s_m + (1-\lambda_v)s_j) - \sigma_j^2)^2] = \sum_{j=1}^p [\lambda_v^2 Var(s_m) + (1-\lambda_v)^2 Var(s_j) + 2\lambda_v(1-\lambda_v)Cov(s_j, s_m) + (\lambda_v E(s_m - s_j) + (E(s_j) - \sigma_j^2))^2]$. Risolvendo analiticamente l'equazione tra zero e la derivata prima di $EQM(s^\star)$ rispetto a $\lambda_v$ si ottiene il valore che minimizza tale EQM, che esiste sempre ed è unico e che si ha in corrispondenza di:

$$\lambda_v^\star = \frac{\sum_{j=1}^p [Var(s_j) - Cov(s_m, s_j) - (E(s_j) - \sigma_j^2))E(s_m - s_j)]}{\sum_{j=1}^p E[(s_m - s_j)^2]}.$$

Indicando con $\bar{x_j} = n^{-1} \sum_{i=1}^n x_{ij}$, con $w_{ij} = (x_{ij} - \bar{x_j})^2$, con $\bar{w_j} = n^{-1} \sum_{i=1}^n w_{ij}$,

con $s_j = \frac{n}{n-1}\bar{w_j}$ (l'usuale stimatore non distorto per la varianza di $x_j$), con $Var(s_j) = \frac{n}{(n-1)^3}\sum_{i=1}^{n}(w_{ij}-\bar{w_j})^2$, con $Cov(s_k, s_j) = \sum_{i=1}^{n}(w_{ik}-\bar{w_k})(w_{ij}-\bar{w_j})$ e assumendo che $Cov(s_m, s_j) \to 0$ con $p \to +\infty$ (e quindi in questo contesto con $p$ molto grande viene imposto $Cov(s_m, s_j) = 0$), l'equazione precedente si semplifica in:

$$\lambda_v^\star = \sum_{j=1}^{p} Var(s_j) / \sum_{j=1}^{p}(s_m - s_j)^2. \tag{2.1}$$

L'approccio utilizzato da Schäfer e Strimmer (2005) per ottenere lo stimatore "di shrinking" per gli elementi fuori dalla diagonale della matrice di varianza e covarianza è l'estensione matriciale dell'approccio appena utilizzato per trovare lo stimatore "di shrinking" delle varianze. Indicando quindi con $\Sigma$ la matrice di varianza e covarianza $(n \times p)$ di $X^{tr}$ (la cui diagonale è composta dagli elementi di $\sigma^2$), con $S$ uno stimatore di $\Sigma$ e con $T$ una matrice diagonale tale che $diag(T) = diag(S)$, lo stimatore delle componenti fuori dalla diagonale (che appunto lascia le componenti sulla diagonale di $S$ intatte per costruzione) di $\Sigma$ dato dallo shrinking fra $S$ e $T$ è $S^\star = \lambda_c T + (1-\lambda_c)S$, con $\lambda_c \in [0,1]$. Come in precedenza, assumendo l'esistenza e la finitezza dei primi due momenti di $S$ e di $T$, il parametro di shrinkage $\lambda_c$, che esiste sempre ed è unico, viene ottenuto come soluzione analitica della minimizzazione dell'EQM di $S^\star$, che è: $EQM(S^\star) = E(||S^\star - \Sigma||_F^2) = E(||\lambda_c T + (1-\lambda_c)S - \Sigma||_F^2) = E(\sum_{i=1}^{p}\sum_{j=1}^{p}(\lambda t_{ij} + (1-\lambda)s_{ij} - \sigma_{ij})^2)$; dove, indicando con $A$ una generica matrice $(n \times p)$, con $||A||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{p} a_{ij}^2}$ si indica la norma di Frobenius, che è l'estensione matriciale della norma euclidea. Dato che $Cov(s_{ij}, t_{ij}) = 0$ *per* $i \neq j$ e utilizzando l'usuale stimatore non distorto per $\Sigma$, come in precedenza l'espressione per il valore di $\lambda_c$ che minimizza l'EQM si riduce a:

$$\lambda_c^\star = \sum_{j \neq i} Var(s_{ij}) / \sum_{j \neq i} s_{ij}^2. \tag{2.2}$$

Per evitare problemi nello shrinkage delle covarianze dovuti alla differenza di scala dei vari predittori, una soluzione alternativa alla standardizzazione

o alla normalizzazione dei predittori è quella di effettuare lo shrinkage sulle correlazioni invece che sulle covarianze: per fare ciò basta sostituire, nel membro di destra dell'equazione di $\lambda_c^\star$, $s_{ij}$ con $s_{ij}/\sqrt{s_{ii}s_{jj}}$.

In campioni con poche osservazioni le stime di $\lambda_v^\star$ e di $\lambda_c^\star$, indicate con $\hat{\lambda_v^\star}$ e $\hat{\lambda_c^\star}$, potrebbero cadere fuori dall'intervallo $[0, 1]$; per risolvere questo problema $\hat{\lambda_v^\star}$ e $\hat{\lambda_c^\star}$ vengono sostituite con $\hat{\lambda_v^{\star\star}} = max(0, min(1, \hat{\lambda_v^\star}))$ e con $\hat{\lambda_c^{\star\star}} = max(0, min(1, \hat{\lambda_c^\star}))$ rispettivamente, così da vincolarle nell'intervallo $[0, 1]$.

L'approccio di stima tramite shrinkage proposto Ledoit e Wolf (2003), applicato al contesto genetico da Schäfer e Strimmer (2005) e da Opgen-Rhein e Strimmer (2007), oltre a risolvere il problema della non invertibilità della matrice di varianza e covarianza con $p > n$, fornisce anche una soluzione a tale problema che garantisce il minimo EQM dello stimatore e che, poichè tale minimo viene trovato in maniera analitica, questo approccio è computazionalmente meno intensivo rispetto a metodi quali il *bootstrap*, la convalida incrociata o l'$MCMC$.

Il metodo HDTBC pertanto consiste nel sostituire allo stimatore della matrice di varianza e covarianza ($S$), nei primi due passi dell'algoritmo della TBC in Sezione 1.3.2 (per ogni osservazione dell'insieme di stima), lo stimatore "di shrinkage" ($S^{\star\star}$); questo, in particolare, consiste nel sostituire il $j$-esimo elemento della diagonale di $S$ (per $j = 1, ..., p$) con $s_j^{\star\star} = \lambda_v^{\star\star} s_m + (1 - \lambda_v^{\star\star}) s_j$ e gli elementi fuori dalla diagonale di $S$ con $S^{\star\star} = \lambda_c^{\star\star} T + (1 - \lambda_c^{\star\star}) S$. Inoltre, l'ulteriore modifica apportata al metodo TBC originale consiste nel confrontare le due statistiche osservate invece che i relativi $\alpha^{oss}$: questo perchè, anche se in teoria la relazione tra le statistiche osservate è biunivoca rispetto alla relazione tra gli $\alpha^{oss}$, in pratica in un contesto di classificazione entrambi gli $\alpha^{oss}$ sono spesso pari a zero (al limite di precisione del calcolatore), mentre le statistiche sono "molto" grandi ma diverse fra loro.

Il metodo, come accennato in precedenza, è stato proposto per estendere la TBC al contesto con $p > n$; inizialmente, prima del metodo basato sullo shrinkage, era stato progettato un metodo basato semplicemente sull'azzeramento degli elementi fuori dala diagonale della matrice di varianza e covarianza della $T^2$ di Hotelling usata dalla TBC: questo metodo, come i metodi proposti in Sezione 2.2 e in Sezione 2.3, non tiene chiaramente in considerazione la struttura di correlazione presente tra i predittori. In seguito, per riuscire a tenere conto di tale correlazione, era stato proposto un metodo basato sulla sostituzione della $T^2$ di Hotelling con una media di (parecchie, e.g 500) $T^2$ di Hotelling calcolate sottocampionando i predittori del dataset (in modo da avere $p < n$) così da rendere invertibile la matrice di varianza e covarianza: sebbene questo metodo possa essere efficace per tenere in considerazione la correlazione fra i predittori, aumenta notevolmente i tempi, già non troppo stretti, richiesti dal calcolatore per effettuare una previsione tramite TBC. Pertanto, oltre alla sicura maggior efficenza computazionale rispetto al metodo appena descritto, si spera che la HDTBC possa avere performance predittive migliori della IBC in contesti con poche osservazioni e tanti predittori. Inoltre si spera anche che la HDTBC possa avere in generale le performance predittive equiparabili alla *Shrinkage Discriminant Analysis* ($SDA$), che è una modifica dell'LDA (introdotta anch'essa da Strimmer) che utilizza $S^{\star\star}$ al posto di $S$ come stimatore di $\Sigma$, ma che con veramente poche osservazioni (e in particolare con un rapporto $p/n$ molto elevato) la HDTBC possa superare l'SDA: questa speranza è alimentata dalla stessa motivazione per cui la TBC (in contesti con $n > p$), con $n$ piccolo possa sfruttare meglio l'informazione nell'insieme di verifica e performare quindi meglio del'LDA. Anche in questo caso la possibile maggior efficacia della HDTBC rispetto al SDA porta con sé una sicura diminuzione di efficienza computazionale; in particolare il tempo di esecuzione che la HDTBC impiega per effettuare le previsioni su un insieme di verifica di $k$ osservazioni è approssivamente $2k$ volte il tempo che impiega l'SDA per fare le previsioni sullo stesso insieme di verifica.

# Capitolo 3

# Applicazioni e Confronti

## 3.1 Modelli Confrontati

In questo capitolo vengono confrontate le performance predittive di vari modelli, misurate con l'accuratezza calcolata tramite convalida incrociata (a 10 fold), su un insieme di 20 dataset. I modelli confrontati, con la relativa ottimizzazione degli iperparametri, sono:
- *Random forest* utilizzando il pacchetto caret (che qui si appoggia sulla libreria *ranger*), mantenendo fisso il numero di alberi a 500 e ottimizzando, ad ogni passo della convalida incrociata, gli iperparametri relativi al numero di predittori da campionare ad ogni split (*mtry*), al numero minimo di osservazioni che un nodo deve avere (*min.node.size*) e alla regola utilizzata per lo split (*splitrule*), seguendo un approccio di tipo *grid search* utilizzando direttamente il campione *out-of-bag*;
- *K nearest neighbor* utilizzando il pacchetto caret (libreria *class*), basato sulla distanza euclidea e con il voto di maggioranza come regola di classificazione; L'iperparametro relativo al numero di nearest neighbor ($k$) è stato ottimizzato, ad ogni passo della convalida incrociata, tramite un'altra convalida incrociata (convalida incrociata nidificata), seguendo un approccio di

tipo *grid search*;
- GLM binomiale con funzione di legame logistica e con penalizzazione *elastic net* utilizzando il pacchetto caret (libreria *glmnet*) e ottimizzando, ad ogni passo della convalida incrociata, gli iperparametri relativi all'intensità della penalizzazione ($\lambda$) e al compromesso tra *lasso* e *ridge* ($\alpha$) tramite un'altra convalida incrociata seguendo un approccio di tipo *grid search*;
- Support vector machine con kernel lineare utilizzando il pacchetto caret (libreria *e1071*) e ottimizzando, ad ogni passo della convalida incrociata, l'iperparametro di regolarizzazione ($C$) tramite un'altra convalida incrociata seguendo un approccio di tipo *grid search*;
- LDA utilizzando il pacchetto caret (libreria *MASS*) per i dataset con $n > p$, mentre per i dataset con $p > n$ è stata utilizzata l'SDA del pacchetto *sda*;
- Versione della TBC che utilizza il t-test e la $T^2$ di Hotelling per i dataset con $n > p$ (Sezione 1.3.2), mentre per i dataset con $p > n$ è stata utilizzata la HDTBC, esposta in Sezione 2.4. L'algoritmo è stato implementato in *R* ed è riportato in Appendice (Algoritmo 1);
- IBC con ottimizzazione dell'iperparametro relativo al numero di nearest neighbor ($k$) effettuata, ad ogni passo della convalida incrociata, tramite un'altra convalida incrociata seguendo un approccio di tipo *grid search*, implementata in *R* (Appendice, Algoritmo 2).

Per i modelli che utilizzano la convalida incrociata come tecnica per la selezione degli iperparametri è stata scelta, anche in questo caso come in quello per l'ottenimento delle previsioni, una convalida incrociata a 10 fold, tranne che per un dataset con 10 osservazioni per cui la convalida incrociata interna è stata fatta a 9 fold. Inoltre, per quanto riguarda l'iperparametro $k$ in IBC, c'è da aggiungere che He et al. (2019) nella IBC non hanno selezionato il valore ottimale di $k$, ma per ogni dataset analizzato hanno misurato le performance della IBC con quattro valori di $k$ diversi ($k = \{3, 5, 7, 9\}$) fissati a priori; in questo studio invece, principalmente per avere un solo modello di IBC (e non quattro) da confrontare con gli altri modelli, è stato deciso di trattare $k$ come un iperparametro (che può assumere i valori 3, 5, 7 e 9, se

possibili) ed è stato quindi anch'esso ottimizzato tramite convalida incrociata a 10 fold. Infine, come fase di preprocessing antecedente all'applicazione dei vari modelli (ad esclusione dell'LDA, della TBC e della random forest che non ne hanno bisogno), nei dataset in cui le variabili esplicative non erano tutte sulla stessa scala è stata effettuata una normalizzazione dei predittori fra $[0, 1]$, normalizzazione effettuata anche da He et al. (2019) sui 40 dataset che hanno analizzato.

## 3.2 Dataset Utilizzati

Per valutare le performance predittive dei vari modelli sono stati utilizzati 17 dataset, 10 con $n > p$ e 7 con $p > n$, anche se in 2 dataset con $p > n$ è stato fatto più di un confronto (perchè sarebbero dataset per la multiclassificazione) e pertanto in totale sono state effettuate 20 classificazioni binarie differenti, 10 con $n > p$ e 10 con $p > n$. In Tabella 3.1 sono riportate le informazioni principali per sintetizzare le caratteristiche dei vari dataset; in particolare le colonne della Tabella 3.1 corrispondono a:

1. Il nome del dataset, dove per i dataset con confronti multipli tra parentesi vengono riportate le iniziali delle due classi confrontate;
2. La *repository* o la libreria *R* dove è stato reperito il dataset;
3. Il numero di osservazioni presenti nel dataset, dopo eventuali fasi di *pre-processing* (omissioni di osservazioni con valori mancanti, filtraggio di osservazioni ripetute) ;
4. Il numero di osservazioni presenti nelle due classi separatamente;
5. Il numero di variabili esplicative, dopo eventuali fasi di *features selection* (selezione dei geni maggiormente espressi);
6. Il rapporto tra il numero di variabili esplicative e il numero di osservazioni.

Tabella 3.1: Informazioni generali sui dataset utilizzati

| Nome | Repository | NumObs | ObsPerClass | Predittori | Pred/Obs |
|---|---|---|---|---|---|
| autism | recount | 22 | 8\|14 | 300 | 13.636 |
| biopsy | MASS | 683 | 444\|239 | 10 | 0.015 |
| brain (g - m) | Kaggle | 56 | 34\|22 | 600 | 10.714 |
| brain (e - p) | Kaggle | 61 | 46\|15 | 600 | 9.836 |
| brain (p - g) | Kaggle | 49 | 34\|15 | 600 | 12.245 |
| breast (LA - LB) | Kaggle | 59 | 29\|30 | 600 | 10.169 |
| breast (b - h) | Kaggle | 71 | 41\|30 | 600 | 8.451 |
| caesarian | UCI | 80 | 34\|46 | 5 | 0.063 |
| colon | Kaggle | 62 | 40\|22 | 200 | 3.226 |
| echocardiogram | UCI | 131 | 88\|43 | 10 | 0.076 |
| gastro | UCI | 76 | 61\|15 | 931 | 12.25 |
| iris | datasets | 100 | 50\|50 | 4 | 0.04 |
| leukemia | Kaggle | 72 | 47\|25 | 500 | 6.944 |
| mnist27 | dslabs | 1000 | 485\|515 | 2 | 0.002 |
| pima | MASS | 532 | 355\|177 | 7 | 0.091 |
| shuttle | MASS | 256 | 145\|111 | 6 | 0.023 |
| sonar | KEEL | 208 | 111\|97 | 60 | 0.288 |
| titanic | Kaggle | 714 | 424\|290 | 7 | 0.009 |
| trains | UCI | 10 | 5\|5 | 20 | 2 |
| urine | boot | 77 | 44\|33 | 6 | 0.078 |

## 3.3 Risultati Ottenuti

In questa sezione vengono riassunti i risultati dei vari modelli, misurati in termini di accuratezza tramite convalida incrociata, ottenuti sulle venti classificazioni binarie effettuate. In particolare tali risultati sono riassunti in Tabella 3.2, dove la prima colonna (come in Tabella 3.1) contiene i nomi dei dataset utilizzati, mentre le rimanenti sette colonne contengono, in ordine dalla seconda alla settima, i risultati relativi a: random forest, k nearest neighbor, GLM binomiale logistico con penalizzazione elastic net, support vector machine, LDA (SDA per dataset con $p > n$), IBC e TBC (HDTBC per dataset con $p > n$). Dopo la Tabella 3.2 vengono riportati tre grafici (che contengono le informazioni della tabella), prodotti per enfatizzare alcuni aspetti salienti emersi dalle applicazioni dei modelli ai dataset. In particolare con il primo grafico (Figura 3.1) vengono confrontate le performance dell'LDA e dell'SDA con quelle della TBC e dell'HDTBC, con il secondo grafico (Figura 3.2) vengono confrontate le performance della TBC (e della HDTBC), dell'IBC con e senza filtraggio (tramite KNN) e del KNN, mentre con il terzo (Figura 3.3) vengono confrontate le performance della TBC (e della HDTBC) con quelle della random forest, del GLM binomiale con penalizzazione elastic net e del support vector machine.

Tabella 3.2: Accuratezza dei modelli sui dataset utilizzati

| Nome | RF | KNN | EN | SVM | LDA | IBC | TBC |
|---|---|---|---|---|---|---|---|
| autism | 0.727 | 0.636 | 0.727 | 0.773 | 0.773 | 0.636 | 0.864 |
| biopsy | 0.971 | 0.969 | 0.969 | 0.971 | 0.960 | 0.969 | 0.960 |
| brain (g - m) | 0.964 | 0.964 | 0.964 | 0.964 | 0.964 | 0.964 | 0.964 |
| brain (e - p) | 0.918 | 0.934 | 0.934 | 0.934 | 0.934 | 0.951 | 0.934 |
| brain (p - g) | 0.919 | 0.755 | 0.878 | 0.939 | 0.919 | 0.755 | 0.919 |
| breast (LA - LB) | 0.898 | 0.831 | 0.898 | 0.932 | 0.932 | 0.847 | 0.932 |
| breast (b - h) | 0.915 | 0.789 | 0.901 | 0.915 | 0.889 | 0.831 | 0.915 |
| caesarian | 0.525 | 0.650 | 0.638 | 0.588 | 0.625 | 0.613 | 0.663 |
| colon | 0.823 | 0.774 | 0.823 | 0.806 | 0.790 | 0.806 | 0.823 |
| echocardiogram | 0.832 | 0.847 | 0.847 | 0.847 | 0.840 | 0.832 | 0.824 |
| gastro | 0.934 | 0.789 | 0.961 | 0.908 | 0.921 | 0.829 | 0.947 |
| iris | 0.920 | 0.950 | 0.940 | 0.950 | 0.960 | 0.940 | 0.960 |
| leukemia | 0.958 | 0.917 | 0.958 | 0.972 | 0.986 | 0.944 | 0.986 |
| mnist27 | 0.826 | 0.832 | 0.783 | 0.790 | 0.792 | 0.830 | 0.789 |
| pima | 0.774 | 0.742 | 0.769 | 0.788 | 0.774 | 0.765 | 0.769 |
| shuttle | 0.984 | 0.969 | 0.980 | 0.980 | 0.934 | 0.969 | 0.934 |
| sonar | 0.87 | 0.817 | 0.755 | 0.750 | 0.716 | 0.846 | 0.707 |
| titanic | 0.814 | 0.776 | 0.784 | 0.780 | 0.789 | 0.796 | 0.782 |
| trains | 0.800 | 0.100 | 0.700 | 0.800 | 0.800 | 0.500 | 0.800 |
| urine | 0.805 | 0.779 | 0.740 | 0.779 | 0.753 | 0.766 | 0.766 |

In Figura 3.1 viene riportata l'accuratezza (in ordinata) della TBC (o HDTBC, linea nera continua) e dell'LDA (o SDA, linea rossa tratteggiata), per i vari dataset analizzati (in ascissa, punti equispaziati da 1 a 20) ordinati (in ordine crescente) in base al rapporto tra il numero di predittori e il numero di osservazioni. La linea verticale punteggiata indica la suddivisione tra i 10 dataset con $n > p$ (a sinistra) e i 10 dataset con $p > n$ (a destra): questa suddivisione rimarca il fatto che per i 10 dataset con $n > p$ vengono confrontate l'LDA con la TBC, mentre per gli altri 10 dataset vengono confrontate l'SDA con la HDTBC. Per quanto riguarda i dataset con $n > p$ dalla Figura 3.1 si nota una fortissima somiglianza tra i due modelli, con LDA che performa lievemente meglio della TBC quando il rapporto tra $p$ ed $n$ è molto ridotto. Anche per i dataset con $p > n$ si nota una fortissima somiglianza tra la HDTBC e l'SDA, tuttavia, nei due dataset con $p/n$ più elevato, la HDTBC mostra un'accuratezza superiore all'SDA, fatto in linea con le motivazioni dietro alla proposta della TBC.

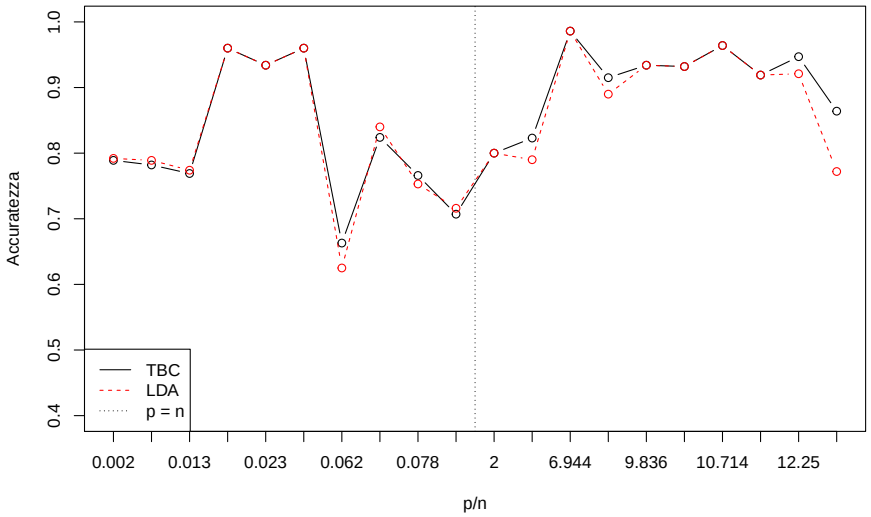


Figura 3.1: Accuratezza di TBC (HDTBC) e LDA (SDA) sui vari dataset analizzati, ordinati in base al rapporto tra il numero di predittori e il numero di osservazioni.

In Figura 3.2 viene riportata l'accuratezza della TBC (o HDTBC, linea nera continua), della IBC ma senza previo filtraggio tramite KNN (risultati non presenti in Tabella 3.2, indicati dalla linea rossa tratteggiata), della IBC con filtraggio tramite KNN (risultati del modello "finale" proposto da He et al. (2019), indicati dalla linea blu tratteggiata) e del KNN (linea gialla tratteggiata); anche qui la linea verticale punteggiata divide i dataset con $n > p$ da quelli con $p > n$. Dalla Figura 3.2 emergono vari fatti interessanti: in primis si nota facilmente come, con i dataset con $p > n$, il modello IBC senza filtraggio abbia ottenuto risultati nettamente inferiori rispetto a tutti gli altri modelli. In secondo luogo si vede come i risultati della IBC (con filtraggio) siano molto più simili al KNN che all'IBC senza filtraggio, rimarcando la discutibilità riguardo l'efficacia delle estensioni della TBC basate sulle IPD. Infine si nota come, con i dataset con $p > n$, la HDTBC abbia performance predittive tendenzialmente superiori rispetto agli altri tre modelli, mostrando quindi come il metodo proposto in Sezione 2.4 abbia prodotto risultati soddisfacenti.

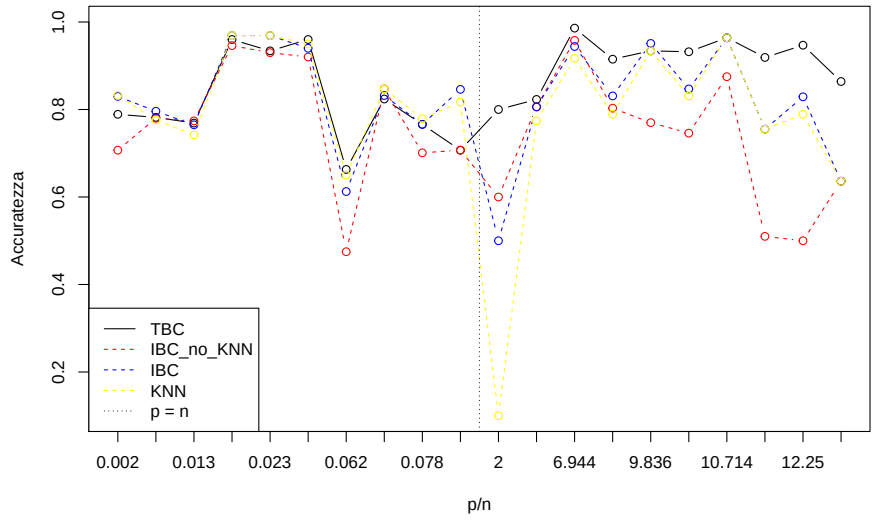


Figura 3.2: Accuratezza di TBC (HDTBC), di IBC senza filtraggio, di IBC con filtraggio e di KNN sui vari dataset analizzati, ordinati in base a $p/n$.

In Figura 3.3 viene riportata l'accuratezza della TBC (o HDTBC, linea nera continua), della random forest (linea rossa tratteggiata), del GLM binomiale con penalizzazione elastic net (linea blu tratteggiata) e del support vector machine (linea gialla tratteggiata); anche qui la linea verticale punteggiata divide i dataset con $n > p$ da quelli con $p > n$. Da questo grafico non emergono esiti eclatanti o straordinari, tuttavia, specialmente riguardo al caso con $p > n$, in generale la HDTBC sembra essere competitiva con i modelli confrontati; questo fatto, anche se andrebbe constatato con molti e molti più casi, costituisce comunque una prima prova a favore dell'efficacia del metodo proposto in questa tesi.

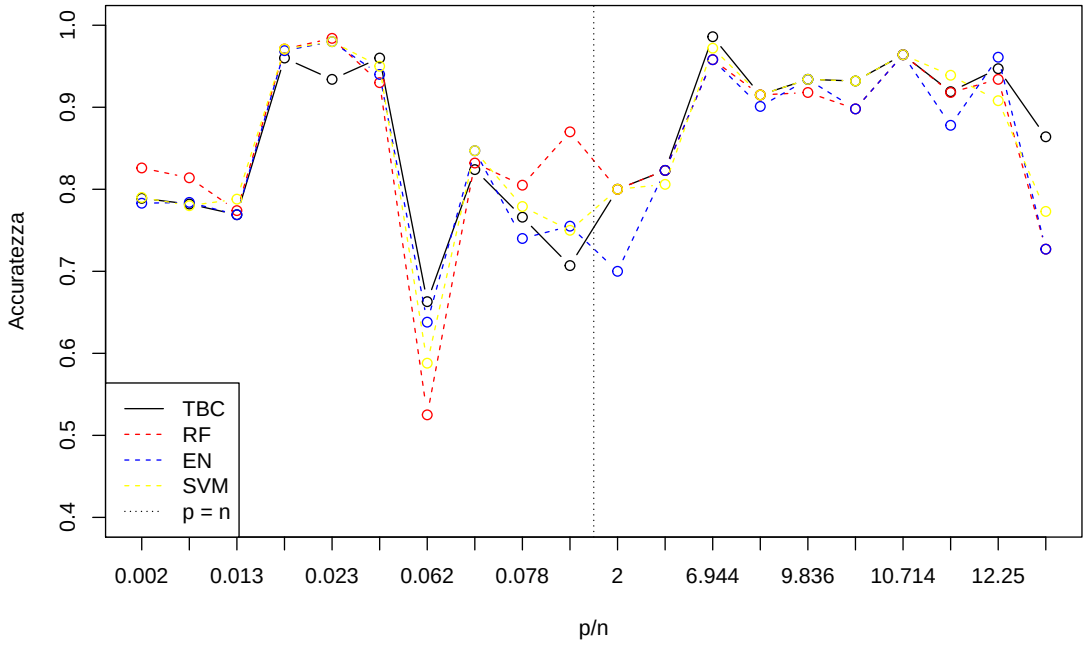


Figura 3.3: Accuratezza di TBC (HDTBC), random forest (RF), GLM binomiale con penalizzazione elastic net (EN) e SVM sui vari dataset analizzati, ordinati in base a $p/n$.

# Conclusione e Spunti Futuri

I risultati ottenuti dalle venti classificazioni binarie effettuate, riassunti in Tabella 3.2 e nelle tre figure del Capitolo 3, hanno mostrato innanzitutto come, riguardo ai dataset analizzati, l'obiettivo principale della tesi sia stato raggiunto, ovvero quello di estendere la TBC al caso con $p > n$ (e $n$ piccolo) in maniera più efficace (in termini di capacità predittive) rispetto alla IBC, che è l'estensione della TBC a $p > n$ proposta da He et al. (2019). In secondo luogo sono state mostrate le forti affinità tra la TBC di Liao e Akritas (2007) e l'LDA nel caso con $n > p$, e le altrettanto forti affinità nel caso con $p > n$ tra la HDTBC e l'SDA (*Shrinkage Discriminant Analysis*, ovvero l'LDA in cui si sostituisce l'usuale stimatore della matrice di varianza e covarianza con quello di "*shrinking*" utilizzato per la HDTBC, proposta anch'essa da Strimmer), anche se nei due dataset con $p/n$ più elevato la HDTBC ha ottenuto performance lievemente superiori rispetto all'SDA. Infine, in generale, sia in termini di efficacia che in termini di efficienza, dai pochi dataset analizzati la TBC sembra essere un metodo di classificazione competitivo con i metodi più famosi (quelli utilizzati nella tesi sono random forest, GLM binomiale con legame logistico e penalizzazione elastic net e support vector machine con kernel lineare), anche se bisognerebbe effettuare uno studio più ampio (prendendo in considerazione molti più dataset) per essere certi dell'effettiva efficacia della TBC (HDTBC).

In futuro come prima cosa si potrebbe fare una modifica alla TBC per tenere in considerazione contesti con classi molto sbilanciate, casi non pre-

si in considerazione in questa tesi: una possibile modifica per gestire questo problema potrebbe consistere nell'utilizzare come regola per la classificazione della TBC, invece che assegnare una nuova osservazione (indicata con $X^{ts}$) alla classe 0 se $\alpha_0^{oss} < \alpha_1^{oss}$, assegnare $X^{ts}$ alla classe 0 se $(1-\ell)\ \alpha_0^{oss} < \ell\ \alpha_1^{oss}$, dove $\ell \in [0,1]$ è una soglia che serve per regolare lo sbilanciamento fra le due classi. Come seconda cosa, dato che in questo lavoro è stato preso in considerazione solamente l'ambito della classificazione binaria, bisognerebbe estendere la TBC alle situazioni con più di due classi; il metodo che hanno proposto Liao e Akritas (2007) per estendere la TBC alla multiclassificazione consiste in una TBC *one vs all* iterativa, ovvero come primo passo vengono effettuate tutte le $c\ell$ (dove con $c\ell > 2$ si indica il numero di classi) classificazioni binarie tra una classe presa singolarmente contro tutte le altre unite assieme (*one vs all* appunto) e viene quindi rimossa la classe con l'$\alpha^{oss}$ più elevato, in seguito vengono effettuate le $c\ell - 1$ classificazioni binarie (*one vs all*) rimaste rimuovendo la classe relativa all'$\alpha^{oss}$ più elevato e si procede in questo modo nella rimozione delle classi finché non ne rimane solamente una, che è la classe dove verrà allocato $X^{ts}$. Un'ulteriore modifica che si potrebbe apportare al metodo TBC sarebbe quella di estenderlo per tenere in considerazione tutta l'informazione dell'insieme di verifica allo stesso tempo e non una sola osservazione alla volta: in breve il metodo, invece che collocare una nuova osservazione alla volta nelle due classi dell'insieme di stima e allocarla (prevederla) nella classe relativa all'$\alpha^{oss}$ più piccolo, consisterebbe nel provare tutte le $2^{n_{ts}}$ (con $n_{ts}$ pari al numero di osservazioni nell'insieme di verifica) disposizioni possibili delle osservazioni dell'insieme di verifica fra le due classi e di allocare tali osservazioni secondo la disposizione che ha generato l'$\alpha^{oss}$ inferiore.

# Appendice A
# Algoritmi implementati in R

## A.1 Test Based Classification

L'algoritmo 1 è l'implementazione in *R* dell'algoritmo di classificazione binaria di Liao e Akritas (2007) (basato sul t-test e sulla $T^2$ di Hotelling) presentato nella Sezione 1.3.2, con l'estensione al caso con $p > n$ (HDTBC) presentata in Sezione 2.4. La funzione *tbc* in input vuole la risposta e i predittori (o il predittore) sull'insieme di stima (indicati con *y_tr* e *x_tr* rispettivamente) e i predittori sull'insieme di verifica (*x_ts*); in output *tbc* ritorna le previsioni per l'insieme di verifica.

**Algoritmo 1:** Test Based Classification

```
tbc ← function(y_tr, x_tr, x_ts){
  # librerie richieste
  lapply(c('purrr', 'Hotelling'), FUN = function(x){
                        do.call("require", list(x)) })
  # oggetti utili per la funzione
  cond_dim ← (NCOL(x_tr) == 1)
  cond_shrink ← (NCOL(x_tr) < NROW(x_tr))
  n_ts ← ifelse(cond_dim, length(x_ts), NROW(x_ts))
  y_tr ← as.factor(y_tr)
```

```
nomi_cl ← levels(y_tr)
cond_dim ← (NCOL(x_tr) == 1)
x_tr ← data.frame(map(x_tr,as.numeric))
x_ts ← data.frame(map(x_ts,as.numeric))
tvals ← rep(0, 2);  prev ← rep(0, n_ts)
# ciclo sulle osservazioni dell'insieme di verifica
for(i in 1:n_ts){
  if(cond_dim){ xtsi ← x_ts[i] }
  else{ xtsi ← x_ts[i,] }
  ddt ← data.frame(yy = y_tr, x_tr)
  # primi due passi dell'algoritmo
  for(cl in 1:2){
    if(cond_dim){
      dati ← data.frame(yy = as.factor(c(cl, y_tr)),
                        c(xtsi, x_tr)) }
    else{
      dati ← data.frame(yy = as.factor(c(cl, y_tr)),
                        rbind(xtsi, x_tr)) }
    db1 ← dati[which(dati$yy == 1),-1]
    db2 ← dati[which(dati$yy == 2),-1]
    # t-test se input unidimensionale
    if(cond_dim){ tvals[cl] ← abs(t.test(db1,db2)$statistic) }
    # T2 di Hotelling se input multidimensionale
    else{
      if(cond_shrink){
        tvals[cl] ← hotelling.stat(db1,db2)$statistic }
      # shrinkage se p > n
      else{ tvals[cl] ← hotelling.stat(db1,db2,
                                shrinkage = TRUE)$statistic }}}
  # terzo passo dell'algoritmo
  prev[i] ← nomi_cl[which.max(tvals)] }

return(prev) }
```

## A.2 Instance Based Classification

L'algoritmo 2 è l'implementazione in *R* dell'algoritmo di classificazione binaria di He et al. (2019) presentato nella Sezione 2.3. Alla funzione *ibc* in input, oltre agli argomenti obbligatori *y_tr*, *x_tr* e *x_ts* come in *tbc*, si può passare un argomento opzionale relativo al filtraggio preliminare delle osservazioni, indicato con *k_nn*: di default tale argomento vale *NULL* (indicando nessun filtraggio delle osservazioni), mentre, se si introduce un valore numerico (accettabile), questo indicherà il numero di osservazioni (per ogni classe) da mantenere ad ogni iterazione dell'algoritmo, osservazioni selezionate appunto tramite il k nearest neighbor. Anche in questo caso in output la funzione ritorna le previsioni per l'insieme di verifica.

**Algoritmo 2:** Instance Based Classification

```
ibc ← function(y_tr, x_tr, x_ts, k_nn = NULL){
  # librerie richieste
  lapply(c('purrr', 'FNN'), FUN = function(x){
    do.call("require", list(x)) })
  # funzione che calcola le IPD tra una matrice ed
  # un vettore tramite distanza euclidea
  ipd ← function(xtr_cl, xts){
    res ← rep(0, NROW(xtr_cl))
    for(i in 1:NROW(xtr_cl)){
      res[i] ← sqrt(sum((xtr_cl[i,] - xts)^2)) }
    return(res) }
  # oggetti utili per la funzione
  cond_dim ← (NCOL(x_tr) == 1)
  n_ts ← ifelse(cond_dim, length(x_ts), NROW(x_ts))
  y_tr ← as.factor(y_tr);  nomi_cl ← levels(y_tr)
  cond_dim ← (NCOL(x_tr) == 1)
  x_tr ← data.frame(map(x_tr,as.numeric))
  x_ts ← data.frame(map(x_ts,as.numeric))
```

```
tvals ← rep(0, 2);  prev ← rep(0, n_ts)
# ciclo sulle osservazioni del test set
for(i in 1:n_ts){
  if(cond_dim){ xtsi ← x_ts[i] }
  else{ xtsi ← x_ts[i,] }
  # senza filtraggio osservazioni tramite knn
  if(is.null(k_nn)){
    dbtr ← data.frame(yy = y_tr, x_tr)
    dist1 ← ipd(dbtr[dbtr$yy==nomi_cl[1],-1],xtsi)
    dist2 ← ipd(dbtr[dbtr$yy==nomi_cl[2],-1],xtsi) }
  # con filtraggio osservazioni tramite knn
  else{
    dbtr ← data.frame(yy = y_tr, x_tr)
    dd1 ← dbtr[dbtr$yy == nomi_cl[1],]
    dd2 ← dbtr[dbtr$yy == nomi_cl[2],]
    k1 ← attr(knn(train=dd1[,-1],test=xtsi,
                  cl=dd1[,1],k=k_nn),'nn.index')[1,]
    k2 ← attr(knn(train=dd2[,-1],test=xtsi,
                  cl=dd2[,1],k=k_nn),'nn.index')[1,]
    if(cond_dim){
      dbknn ← data.frame(yy = c(dd1[k1,1],dd2[k2,1]),
                         c(dd1[k1,-1],dd2[k2,-1])) }
    if(!cond_dim){
      dbknn ← data.frame(yy = c(dd1[k1,1],dd2[k2,1]),
                         rbind(dd1[k1,-1],dd2[k2,-1]))
      dist1 ← ipd(dbknn[dbknn$yy==1,-1],xtsi)
      dist2 ← ipd(dbknn[dbknn$yy==2,-1],xtsi) }}
  # primi due passi dell'algoritmo
  p1 ← wilcox.test(dist1,dist2,alternative='less')$p.value
  p2 ← wilcox.test(dist1,dist2,alternative='greater')$p.value
  # terzo passo dell'algoritmo
  prev[i] ← ifelse(p1 < p2, nomi_cl[1], nomi_cl[2]) }
return(prev) }
```

# Bibliografia


Alcalá-Fdez, Jesús et al. (2011). "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework." In: *Journal of Multiple-Valued Logic & Soft Computing* 17.

Asuncion, Arthur e David Newman (2007). *UCI machine learning repository.*

Baringhaus, Ludwig e Carsten Franz (2004). "On a new multivariate two-sample test". In: *Journal of multivariate analysis* 88.1, pp. 190–206.

Ghimire, Santosh e Haiyan Wang (2012). "Classification of image pixels based on minimum distance and hypothesis testing". In: *Computational Statistics & Data Analysis* 56.7, pp. 2273–2287.

Guo, Lingzhe e Reza Modarres (2019). "Interpoint distance classification of high dimensional discrete observations". In: *International Statistical Review* 87.2, pp. 191–206.

He, Zengyou et al. (2019). "Instance-based classification through hypothesis testing". In: *arXiv preprint arXiv:1901.00560.*

Ledoit, Olivier e Michael Wolf (2003). "Improved estimation of the covariance matrix of stock returns with an application to portfolio selection". In: *Journal of empirical finance* 10.5, pp. 603–621.

Liao, Shu-Min e Michael Akritas (2007). "Test-based classification: A linkage between classification and statistical testing". In: *Statistics & probability letters* 77.12, pp. 1269–1281.

Opgen-Rhein, Rainer e Korbinian Strimmer (2007). "Accurate ranking of differentially expressed genes by a distribution-free shrinkage approach." In: *Statistical Applications in Genetics & Molecular Biology* 6.1.

Schäfer, Juliane e Korbinian Strimmer (2005). “A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics”. In: *Statistical applications in genetics and molecular biology* 4.1.