UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di
Economia e Management

Corso di Laurea in
Statistica

Metodi di classificazione basati su alberi: applicazioni empiriche in R

Relatore

Agnese Panzera

Candidato

Leonardo Michi

Anno Accademico 2022/2023

Alla famiglia che mi ha supportato durante il percorso.

A mio nonno che mi ha trasmesso la sua felicità.

INDICE

Introduzione

La classificazione statistica è un concetto fondamentale nel campo dell'analisi dei dati ed è utilizzata nei processi decisionali di tutte le industrie di qualsiasi settore. La tesi è strutturata in tre capitoli, ognuno dei quali affronta aspetti distinti della classificazione e le sue applicazioni. Nel primo capitolo, approfondiremo i concetti fondamentali, differenziando tra classificazione e regressione. Inoltre, esploreremo tre metodi di classificazione parametrica tra i più utilizzati: regressione logistica, analisi discriminante lineare (LDA) e Naive Bayes. Alla fine, saranno introdotti i metodi di valutazione dei modelli che saranno fondamentali nella fase di applicazione per valutarne la bontà.

Il secondo capitolo sposta l'attenzione sui metodi di classificazione basati sugli alberi. Questi algoritmi, come Decision Trees, Random Forests, Bagging e Boosting, hanno guadagnato enorme popolarità per la loro capacità di gestire dati complessi e catturare relazioni non lineari. Approfondiremo la meccanica di questi metodi, illustrandone i punti di forza e le potenziali insidie.

Il terzo e ultimo capitolo di questa tesi si concentrata su prove empiriche, utilizzando set di dati provenienti da diversi settori. Questi test empirici mirano a convalidare l'efficacia dei metodi di classificazione discussi nei capitoli precedenti negli scenari del mondo reale. I set di dati selezionati coprono vari settori e sarà svolta un'analisi comparativa per assistere nella selezione dell'algoritmo ad albero più adatto per compiti di classificazione specifici.

In conclusione, questa tesi cerca di fornire ai lettori una comprensione completa della classificazione e dei suoi vari metodi, dai modelli parametrici agli algoritmi ad albero.

1. Classificazione

La classificazione è il processo di identificazione e raggruppamento di oggetti, persone, idee in categorie predeterminate. In ambito statistico consiste nell'utilizzare specifiche tecniche di classificazione, o *classifier,* e, grazie ad esse, raggruppare le singole osservazioni che presentano *pattern* similari. In questo capitolo presenteremo la *regressione lineare*, le principali differenze tra essa e la classificazione fino ad arrivare alle tecniche di classificazione, parametriche e non parametriche, più diffuse.

1.1 Regressione lineare

La regressione è un metodo per studiare la relazione tra una variabile di risposta Y e una covariata X anche detta variabile esplicativa. Un modo per esprimere la relazione tra Y e X è tramite la *funzione di regressione:*

$$r(x) = E(Y|X = x)$$

L'obiettivo è quello di stimare la funzione dai dati $(Y_1, \boldsymbol{X}_1), \dots, (Y_n, \boldsymbol{X}_n)$.

Il modello di regressione lineare [7] è il seguente:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_k X_{ik} + \varepsilon_i$$

In cui:

- X_{ij} rappresenta la i-esima osservazione relativa alla j-esima variabile esplicativa
- Il coefficiente β_j esprime la variazione della variabile di risposta all'aumento unitario di X_j e a parità di tutte le altre variabili esplicative
- ε_i sono gli errori con media zero, varianza costante e tra loro indipendenti.

Il modello è esprimibile, grazie all'algebra delle matrici, con una singola equazione:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$$

In cui:

- $\boldsymbol{Y}$ è il vettore $nx1$ delle osservazioni sulla risposta;
- $\boldsymbol{X}$ è la matrice $nxk + 1$ del modello;

- $\boldsymbol{\beta}$ è il vettore $kx1$ dei parametri di regressione;
- $\boldsymbol{\varepsilon}$ è il vettore $nx1$ degli errori

I coefficienti di regressione sono sconosciuti e devono essere stimati. Ciò avviene attraverso il criterio dei minimi quadrati che raccomanda di stimare i parametri incogniti $\beta_0, \dots, \beta_k$ con i valori $\widehat{\beta_0}, \dots, \widehat{\beta_k}$ che minimizzano la funzione di perdita:

$$\| \boldsymbol{Y} - \boldsymbol{X\beta} \|^2$$

rispetto a $\boldsymbol{\beta}$.

Gli stimatori dei minimi quadrati $\widehat{\boldsymbol{\beta}}$ esistono, sono unici ed uguali a:

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

1.2 Classificazione e regressione

Il modello di regressione lineare precedentemente descritto è usato nel caso in cui la variabile di risposta sia quantitativa continua. Al contrario, nel caso in cui la variabile di risposta sia qualitativa, o *categorica,* entrano in gioco le tecniche di classificazione. Queste tecniche stimano la probabilità che l'osservazione appartenga a una delle categorie della variabile qualitativa in base a variabili esplicative, sia qualitative che quantitative. Un esempio di classificazione è l'attribuzione di un fiore a una certa specie in base al colore, forma, lunghezza dei petali.

Così come nella regressione, nella classificazione è fondamentale un *training set* di osservazioni$(x_1, y_1), \dots, (x_n, y_n)$, ovvero un gruppo di osservazioni utilizzate per addestrare il modello*,* che verrà poi valutato su un *test set,* un gruppo di osservazioni sulle quali non è stato addestrato un modello e che servirà per valutare la bontà della stima.

La motivazione per cui la regressione lineare non è appropriata [1] per le risposte qualitative è molto semplice, come mostrato dalla *Figura 1*. Nel caso in cui fosse utilizzata, le variabili di risposta dovrebbero essere codificate numericamente, per esempio $Y = 1\ o\ 2\ o\ 3$. Con questa codifica, il criterio dei minimi quadrati potrebbe essere utilizzato per stimare i coefficienti del modello *Y.*

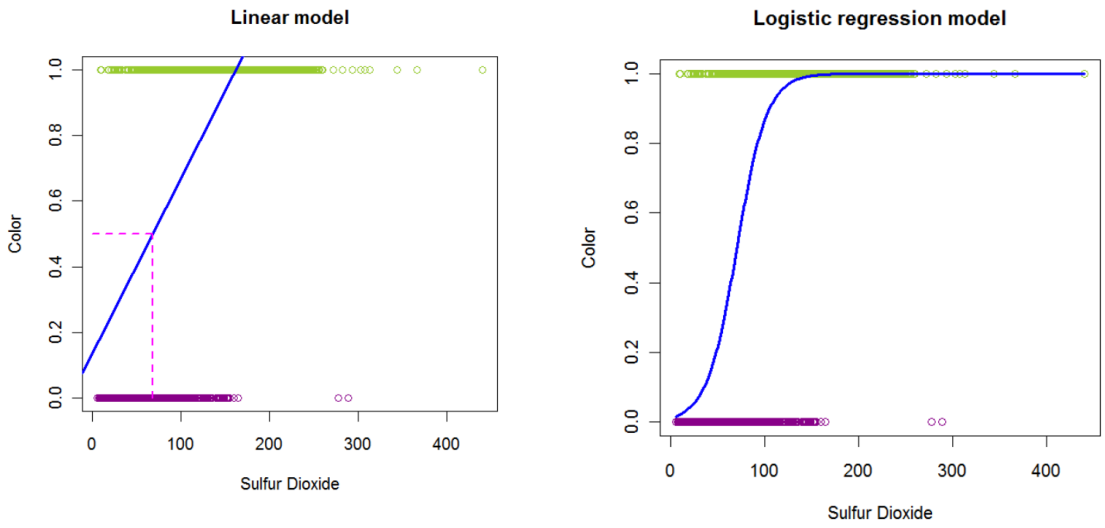

figura 1: confronto tra regressione lineare e regressione logistica

Il problema è che dovremmo assumere un preciso ordine tra le variabili e che la distanza tra le variabili $Y = 1$ *e* $Y = 2$ è la stessa tra $Y = 2$ *e* $Y = 3$, cosa che, nel caso di variabili qualitative, è altamente improbabile. Inoltre, nel caso in cui l'assegnazione delle variabili numeriche cambiasse con una delle possibili combinazioni, la nuova codifica produrrebbe un diverso modello lineare con diverse stime, *figura 1*. Per questo è preferibile l'uso di metodi di classificazione come la regressione logistica, l'analisi discriminante lineare, Naive Bayes.

1.3 Regressione Logistica

1.3.1 Il modello logistico lineare

Nel caso di variabili di risposta binaria, la regressione lineare comporta una serie di problemi che possono essere superati utilizzando una funzione che, date le coppie di valori (x_i, y_i) per $i = 1, \dots, n$ su una variabile esplicativa X e su una risposta binaria Y, fornisce *outputs* compresi tra 0 e 1 per tutti i valori di X. Nella regressione logistica [7] la funzione è:

$$p_i = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

Il modello logistico lineare si basa sulle seguenti ipotesi:

- Le osservazioni y_i sono realizzazioni delle variabili Y_i.

- Che quest'ultime abbiano distribuzioni Bernoulli di parametri p_i, in cui $p_i = P(Y_i = 1|x_i)$ sono le probabilità di successo.
- Che i logit delle probabilità p_i siano funzioni lineari della variabile esplicativa $logit(p_i) = \beta_0 + \beta_1 x_i$.

1.3.2 Stima dei coefficienti di regressione

I coefficienti $\beta_0 \, e \, \beta_1$ sono sconosciuti e devono essere stimati sulla base dei dati a disposizione. La stima avviene col metodo di massima verosimiglianza. La funzione di verosimiglianza è la seguente:

$$L(\beta_0, \beta_1) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i}$$

Che può anche essere espressa sotto forma di logaritmo:

$$l(\beta_0, \beta_1) = \sum_i y_i logit(p_i) - \sum_i \log\{1 + \exp[logit(p_i)]\}$$

La stima di massima verosimiglianza dei coefficienti si ottiene massimizzando la funzione log-verosimiglianza rispetto a $\beta_0 \, e \, \beta_1$.

L'estensione del modello logistico lineare con una sola variabile esplicativa è abbastanza semplice e si basa sulla definizione di un pre*dittore lineare* che include *k* termini. Il modello può essere quindi generalizzato come segue:

$$logit(p_i) = \log\left(\frac{p_i}{1 - p_i}\right) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}$$

Sotto l'ipotesi che le osservazioni $(Y_1, \ldots, Y_n)$ siano indipendenti e distribuite come Bernoulli con probabilità di successo $p_i = P(Y_i = 1|X = x)$.

1.4 Metodi basati sul teorema di Bayes

Esistono metodi che utilizzano un diverso approccio rispetto al modello di regressione logistica [1]. Modellando la distribuzione dei predittori di X separatamente in ognuna

delle classi di risposta e con l'utilizzo del teorema di Bayes è stimata:

$$P(Y = k|X = x)$$

Il motivo principale per il quale vengono utilizzati altri metodi oltre alla regressione logistica è che questi approcci possono essere estesi più semplicemente ai casi con più di due classi di risposta. Supponiamo di voler classificare un'osservazione in una delle *K* classi dove $K \geq 2$. Indichiamo con π_k la probabilità a priori che un'osservazione casuale appartenga a una specifica classe *k,* per cui la *funzione di densità* di *X* per tale osservazione sia:

$$f_k(X) \equiv P(X|Y = k)$$

$f_k(x)$ sarà grande se c'è un'alta probabilità che l'osservazione di una classe *k è* $X = x$*;* al contrario, sarà piccola se la probabilità che $X = x$ è molto bassa. Ciò può essere riassunto con la formula di Bayes:

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^{K} \pi_i f_i(x)}$$

1.4.1 Analisi discriminante lineare

Con l'assunzione di avere un solo predittore, l'obiettivo è ottenere la stima di $f_k(x)$ per stimare $p_k(x)$ e poi classificare ogni osservazione alla classe in cui $p_k(x)$ è maggiore. La stima è ottenuta sotto l'assunzione che $f_k(x)$ sia *normale*, ovvero:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2\sigma_k^2}(x-\mu_k)^2}$$

Facendo il logaritmo e sistemando i termini otteniamo che l'osservazione è assegnata alla classe per la quale:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

è maggiore.

Per stimare i parametri $\mu_1, \dots \mu_{K,} \pi_1, \dots, \pi_K, \sigma^2$, l'analisi discriminante lineare utilizza le seguenti stime:

$$\widehat{\mu_k} = \frac{1}{n_k}\sum_{i:y_i=k} x_i \text{ e } \widehat{\sigma^2} = \frac{1}{n-K}\sum_{k=1}\sum_{i:y_i=k}(x_i - \widehat{u_k})^2,$$

dove *n* è il numero totale di osservazioni del *training set* e n_kè il numero di osservazioni del training set in ciascuna classe *k.*

L'analisi discriminante lineare può essere estesa pure al caso di più predittori, con l'assunzione che $X = (X_1, X_2, \dots, X_p)$ è scelto da una *distribuzione normale multivariata* e quindi anche le osservazioni di una specifica classe sono scelte da una distribuzione normale multivariata $N(u_k, \Sigma)$ dove u_k è il vettore delle media di classe e Σ è la matrice delle covarianze comune a tutte le classi.

1.4.2 Naive Bayes

Per usare la formula di Bayes in pratica è necessario stimare la probabilità a priori $\pi_1, \dots, \pi_K$ e le funzioni di densità $f_1(x), \dots, f_K(x)$. Stimare le probabilità a priori è relativamente facile, poiché è possibile stimarle come la proporzione delle osservazioni del training set appartenenti a una specifica classe *k.* al contrario, la stima delle funzioni di densità richiede una maggiore attenzione. Nell'analisi discriminante lineare era utilizzata l'assunzione di normalità della funzione di densità che semplifica notevolmente il processo. Il classificatore Naive Bayes, invece di assumere che tali funzioni appartengano a una specifica distribuzione, si basa su una diversa assunzione, ovvero che all'interno di ogni classe *k,* i *p* predittori sono indipendenti. Ciò significa che:

$$f_k(x) = f_{k1}(x_1) \cdot f_{k2}(x_2) \cdot \dots \cdot f_{kp}(x_p)$$

Dove f_{kj}è la funzione di densità del j-esimo predittore tra le osservazioni nella k-esima classe. Sostituendo la funzione ottenuta dall'assunzione nella formula del Naive Bayes l'espressione per la probabilità a posteriori diventa:

$$P(Y = k|X = x) = \frac{\pi_k \cdot f_{k1}(x_1) \cdot f_{k2}(x_2) \cdot \dots \cdot f_{kp}(x_p)}{\sum_{i=1}^{K} \pi_i \cdot f_{i1}(x_1) \cdot f_{i2}(x_2) \cdot \dots \cdot f_{ip}(x_p)}$$

Se la probabilità risultante che l'osservazione appartenga alla classe k è maggiore di 0.5, allora verrà assegnata a quella classe mentre se è minore all'altra classe.

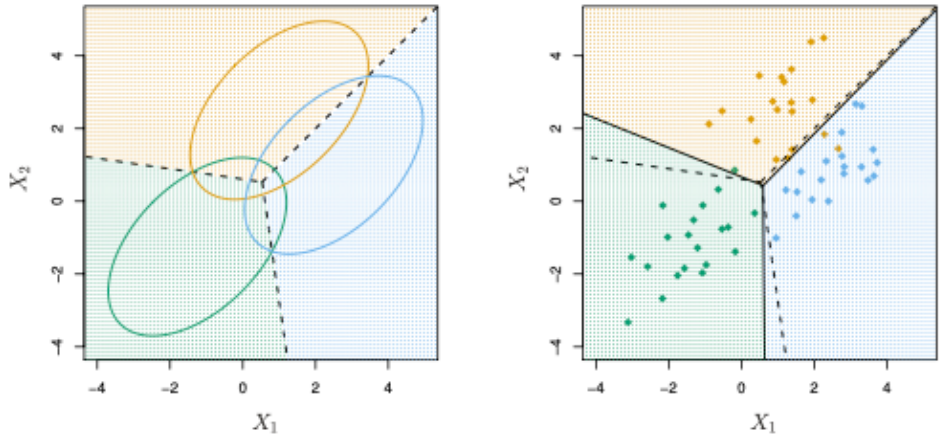

Figura 2: confronto di esempio con tre classi e due predittori tra LDA e Naive Bayes

Nessuno di questi metodi parametrici di classificazione prevale costantemente sugli altri ma dipende dai singoli casi. Quando i veri confini decisionali sono lineari, l'analisi discriminate lineare e la regressione logistica tendono a performare meglio, mentre quando sono moderatamente non lineari, Naive Bayes prevale, *figura 2*. Quando i confini decisionali sono molto complicati un approccio non parametrico come i *metodi basati su alberi* può essere superiore.

1.5 Model assessment

La prestazione di un modello di classificazione dipende dalla sua capacità di stimare correttamente le osservazioni del test set [2]. La valutazione della prestazione è molto importante poiché da essa dipende la scelta di uno specifico modello. Per quanto riguarda la regressione Y è la variabile di risposta, $\boldsymbol{X}$ è il vettore delle variabili casuali e $\hat{f}(\boldsymbol{X})$ è il modello di previsione stimato da un training set T. La funzione di perdita per misurare l'errore tra Y e $\hat{f}(\boldsymbol{X})$ è $L(Y, \hat{f}(\boldsymbol{X}))$. Solitamente sono utilizzati l'errore quadratico o assoluto:

$$L\left(Y, \hat{f}(\boldsymbol{X})\right) = \begin{cases} (Y - \hat{f}(\boldsymbol{X}))^2 \\ |Y - \hat{f}(\boldsymbol{X})| \end{cases}$$

il *test error* è l'errore di stima su un campione indipendente dove $\boldsymbol{X}$ e Y sono scelti in maniera casuale dalla popolazione ed equivale a:

$$Err_T = E[L(Y, \hat{f}(\boldsymbol{X})|T]$$

Un'altra misura collegata ad essa è il *test error atteso*:

$$Err = E[L\big(Y, \hat{f}(\boldsymbol{X})\big] = E[Err_T]$$

Per quanto riguarda la classificazione le funzioni di perdita sono differenti. Considerando una variabile qualitativa $G \in \boldsymbol{G}$ con G che va da 1 *a* K la probabilità $p_k(\boldsymbol{X}) = p(G = k|\boldsymbol{X})$ e quindi $\hat{G}(\boldsymbol{X}) = argmax_k \hat{p}_k(\boldsymbol{x})$. Le funzioni di perdita più utilizzate per la classificazione sono:

$$L(G, \hat{G}(\boldsymbol{X})) = I(G \neq \hat{G}(x)$$

$$L\big(G, \hat{p}(\boldsymbol{X})\big) = -2 \sum_{k=1}^{K} I(G = k) \log \hat{p}_k(\boldsymbol{X})$$

In questo caso il test error è:

$$Err_T = E\left[L\left(G, \hat{G}(\boldsymbol{X})\right)\middle|T\right]$$

Nei problemi di classificazione è possibile visualizzare la prestazione di un modello graficamente grazie alla *matrice di confusione* che è una tabella $2x2$ in cui, *Vero positivo* indica il numero di osservazioni classificate correttamente come positive, *Vero negativo* indica il numero di osservazioni classificate correttamente come negative, *Falso positivo* indica il numero di osservazioni non classificate correttamente come positive e *falso negativo* le osservazioni classificate non correttamente come negative.

		previsto	
	Popolazione totale (PT)	positivo	negativo
Effettivo	positivo	Vero positivo (VP)	Falso negativo (FN)
	negativo	Falso positivo (FP)	Vero negativo (VN)

Oltre alla misura di errore di classificazione del modello calcolata come: $\frac{FP+FN}{PT}$ è possibile calcolare altre misure che derivano dalla tabella:

- $\frac{VP+VN}{PT}$ *accuratezza*
- $\frac{VP}{VP+FN}$ *sensitività*
- $\frac{VN}{VN+FP}$ *specificità*
- $\frac{VP}{VP+FP}$ *precisione*
- $\frac{2VP}{2VP+FP+FN}$ *F1*

1.5.1 Cross-validation

La cross-validation è con molta probabilità il metodo più usato per calcolare l'errore di stima. Questo metodo stima direttamente l'errore extra-campione atteso $Err = E[L\left(\boldsymbol{Y}, \hat{f}(\boldsymbol{X})\right)]$, l'errore di generalizzazione medio quando $\hat{f}(\boldsymbol{X})$ è applicato a un campione test indipendente della distribuzione congiunta di $\boldsymbol{X}$ *e* $\boldsymbol{Y}$, dove $L\left(\boldsymbol{Y}, \hat{f}(\boldsymbol{X})\right)$ è la funzione di perdita.

L'ideale è creare un *validation set* a parte per calcolare le prestazioni del modello di predizione. Siccome i dati sono spesso pochi si rende necessario l'utilizzo della *K-fold cross-validation,* la quale usa parte dei dati disponibili per addestrare il modello e un'altra parte per testarlo. I dati sono divisi in *K* parti uguali. Per la *k*-esima parte, si addestra il modello sulle altre *k-1* parti e si calcola l'errore di stima del modello addestrato quando si stima la *k-esima* parte dei dati. Si ripete il processo per *k=1, 2, ..., K* e si combina i *k* errori di stima. Formalmente [2]:

$$CV(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{f}^{-k(i)}(x_i)\right)$$

dove $\hat{f}^{-k}(x)$ è la funzione addestrata, calcolata con la *k*-esima parte dei dati rimossi.

1.5.2 Bootstrap

i metodi di ricampionamento consistono nell'estrarre ripetutamente campioni dalle osservazioni di un training set e addestrare nuovamente il modello su ognuno di essi per ottenere informazioni aggiuntive sul modello stesso. Uno dei metodi di ri-campionamento più importante è il *bootstrap* che sarà indispensabili ai fini degli alberi decisionali.

Il bootstrap è un metodo per stimare gli errori standard e gli intervalli di confidenza[1]. Come la cross-validation, il bootstrap mira a calcolare l'errore di stima Err_T ma tipicamente stima bene solo l'errore di stima atteso. Supponiamo di avere un modello da addestrare sui dati di un training set $\boldsymbol{Z} = (z_1, z_2, \dots, z_N)$ dove $z_i = (x_i, y_i)$. L'idea di base è quella di formare casualmente dei datasets con ripetizione dalle osservazioni del training set. Questo è fatto B volte, creando B campioni bootstrap. Il modello viene nuovamente addestrato su ognuno di essi. Dai campionamenti può essere stimano ogni aspetto della distribuzione $S(\boldsymbol{Z})$ che è una qualsiasi quantità calcolata dai dati $\boldsymbol{Z}$, come la varianza:

$$\widehat{Var}[S(z)] = \frac{1}{B-1}\sum_{b=1}^{B}(S(Z^{*b}) - \bar{S}^{*})^2$$

dove $\bar{S}^{*} = \sum_b S(Z^{*b})/B$.

Per calcolare l'errore di stima, un possibile metodo è di addestrare il modello su dei campionamenti bootstrap e poi tenere conto di come stima bene il training set originale. Se $\hat{f}^{*b}(x_i)$ è il valore stimato a x_i, allora:

$$\widehat{Err}_{boot} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L(y_i, \hat{f}^{*b}(x_i))$$

Questa stima, però, non da buoni risultati in generale poiché i campioni bootstrap si comportano come campioni training, mentre i training set originali si comportano come i test set e questi due campioni hanno osservazioni in comune. Questa sovrapposizione può portare a overfitting. È possibile ottenere un miglior stimatore bootstrap tenendo conto, per ogni osservazione, delle stime che provengono dai campionamenti bootstrap che non contengono tale osservazione. L'errore di stima bootstrap *Leave-one-out* è:

$$\widehat{Err}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}^{*b}(x_i))$$

dove C^{-i} sono gli indici dei campionamenti bootstrap che non contengono l'osservazione i e $\left|C^{-i}\right|$ è il numero di questi campionamenti.

2. Metodi di classificazione basati su alberi

I metodi basati su alberi sono una serie di algoritmi non parametrici per la regressione e la classificazione. L Breiman è considerato il pioniere di tali metodi e nel 2001 introduce l'algoritmo *random forest*[4] che ha mostrato ottime *performance* quando il numero di variabili è particolarmente elevato. Inoltre, essendo molto versatile è adatto a problemi su larga scala. Questo algoritmo opera basandosi sul semplice ma efficace principio *"dividi e conquista"* che consiste nel dividere ricorsivamente un problema in due sotto-problemi finché non diventano abbastanza semplici da essere risolti direttamente.

In questo secondo capitolo sono presentati i principi base degli alberi decisionali, le tecniche di *bagging, boosting* ed è descritto il processo di costruzione delle random forest.

2.1 Alberi decisionali

Gli alberi decisionali possono essere usati sia per la regressione che per la classificazione ed hanno un processo di costruzione similare.

2.1.1 Costruzione di un albero di regressione

Partendo da un albero di regressione, la sua costruzione si può banalmente riassumere in due passaggi.

- Inizialmente lo spazio dei predittori χ , ovvero, i possibili valori che $X_1, X_2, \dots, X_p$ possono assumere, è diviso in $R_1, R_2, \dots R_J$ regioni non sovrapponibili che formano una partizione di χ.
- Per ogni osservazione che cade in una certa regione R_j, viene effettuata una stima, che è semplicemente la media dei valori della variabile di risposta ottenuta sulle osservazioni del *training set* in R_j.

Chiaramente è necessario costruire le regioni, che potrebbero avere qualunque forma, ma lo spazio dei predittori è generalmente diviso in *iperrettangoli*. L'obiettivo è quello di trovare gli iperretangoli $R_1, R_2, \dots R_J$ che minimizzano la somma dei quadrati residui:

$$\sum_{j=1}^{J}\sum_{i\in R_j}(y_i-\hat{y}_{R_j})^2$$

Dove $\hat{y}_{R_j}$ è la media delle variabili di risposta del j-esimo iperrettangolo, ottenuta sulle osservazioni del training set.

Per trovare le partizioni di χ si rende necessario l'utilizzo di un algoritmo di divisione binaria ricorsiva, con una ricerca *greedy e top-down* [1]*, figura 3.*

a. L'algoritmo è *top-down* poiché inizia dalla cima dell'albero, quando tutte le osservazioni appartengono a una sola regione, e successivamente la divide.

b. L'algoritmo è *binario* perché ogni partizione è successivamente divisa in ulteriori due partizioni, fino alla fine dell'albero.

c. L'algoritmo è *greedy* perché è ad ogni passaggio che è scelta la migliore divisione.

L'algoritmo, inizialmente, sceglie un predittore X_j e il valore *s* che divide lo spazio dei predittori χ negli iperrettangoli $\{X|X_j<s\}$ e $\{X|X_j\geq s\}$ che minimizzano la somma dei quadrati residui. Per trovare X_j e *s* si considerano tutti i predittori e tutti i valori che *s* può assumere per ogni predittore, ovvero per ogni *j e s* definiamo la coppia di semipiani:

$$R_1(j,s)=\{X|X_j<s\}\ \ e\ \ R_2(j,s)=\{X|X_j\geq s\}$$

e troviamo i valori *j* e *s* che minimizzano l'equazione:

$$\sum_{i:x_i\in R_1(j,s)}(y_i-\hat{y}_{R_1})^2+\sum_{i:x_i\in R_2(j,s)}(y_i-\hat{y}_{R_2})^2$$

Successivamente si ripete il processo su ciascuno dei nuovi semipiani individuati finché lo *stopping* criterion non è raggiunto e sono state definite tutte le regioni.

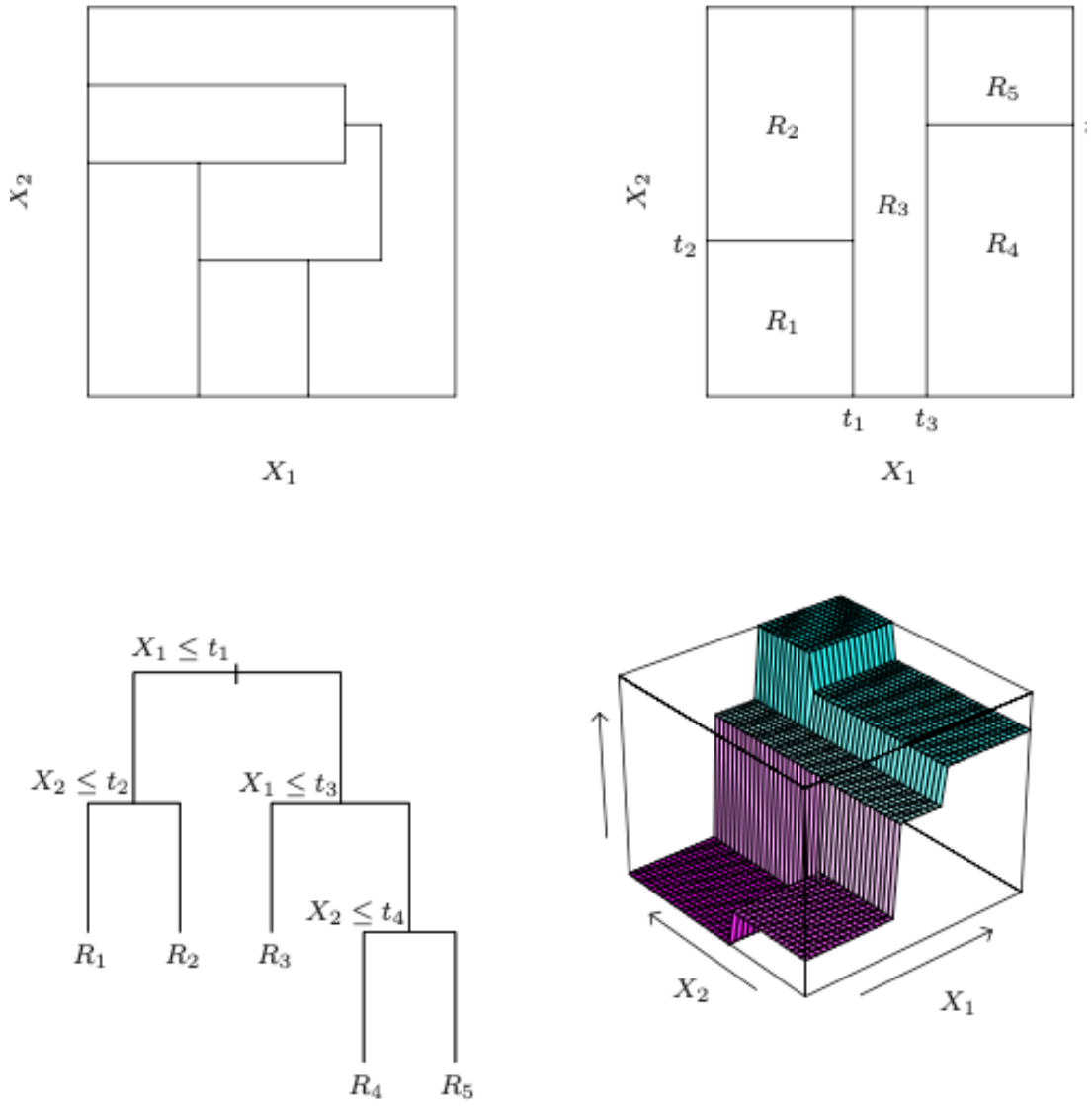

Figura 3. In alto confronto tra una partizione di uno spazio bi-dimensionale ottenuto con la divisione binaria ricorsiva e senza. In basso a sinistra rappresentazione ad albero della partizione in alto a destra. In basso a destra un grafico 3D della superfice di stima.

2.1.2 Tree pruning

Tree pruning significa letteralmente "potare l'albero". La scelta della grandezza di un albero è, infatti, un elemento fondamentale per le tecniche basate sugli alberi decisionali. Un albero molto grande potrebbe portare ad un *overfitting* dei dati, mentre un albero troppo piccolo non è in grado di catturare buona parte della varianza. La grandezza dell'albero è un *tuning parameter,* che governa la complessità del modello, e perciò una scelta accurata di tale parametro si rivela fondamentale per avere il giusto *trade-off* ed ottenere così una varianza comunque non elevata ed evitare l'overfitting, oltre ad un miglioramento dell'interpretabilità.

Il processo consiste nel far crescere un grande albero T_0, arrestando il processo di divisione quando viene raggiunta la dimensione minima del nodo, scelta a priori. L'albero è *potato* con l'uso della *cost-complexity pruning* [2]:

- Definisco un sottoalbero $T \subset T_0$ che può essere ognuno degli alberi ottenuti eliminando un qualunque numero dei suoi nodi interni.
- Indicizzo *i* nodi *j* che rappresentano la regione R_j
- Denoto con $|T|$ il numero di nodi terminali in T
- Definisco il criterio di complessità dei costi:

$$C_a(T) = \sum_{j=1}^{|T|} N_j Q_j(T) + \alpha|T|$$

dove:

- $N_j = \#\{x_i \in R_j\}$
- $\hat{c}_j = \frac{1}{N_j}\sum_{x_i \in R_j} y_i$
- $Q_j(T) = \frac{1}{N_j}\sum_{x_i \in R_j}(y_i - \hat{c}_j)^2$

Per ogni valore di α, l'obiettivo è trovare l'albero *potato* $T_\alpha \subset T_0$ che ha il minore $C_a(T)$. Il parametro $\alpha \geq 0$, infatti, controlla il giusto compromesso tra grandezza dell'albero e la sua bontà di adattamento ai dati. Più grande è α e più piccolo sarà l'albero $T(\alpha)$ e viceversa. Esso è scelto per mezzo della *cross-validation* che individua una serie di valori corrispondenti a una sequenza di alberi. La stima di α è ottenuta con la *fold cross-validation* ed è scelto il valore di $\hat{\alpha}$ che minimizza la cross-validation della devianza. L'albero finale è $T_{\hat{\alpha}}$.

2.1.3 Alberi di classificazione

un albero di classificazione, a differenza di un albero di regressione è utilizzato per stimare la corretta classe di appartenenza di una variabile di risposta qualitativa. In poche parole, assegna ogni osservazione alla classe più comune delle osservazioni del training set nella ragione a cui appartiene.

Per costruire l'albero possiamo usare il solito algoritmo usato per la regressione ma con alcune modifiche:

- una diversa scelta del predittore
- un nuovo criterio di divisione binaria

La scelta del predittore è data dalla teoria[1]: le unità in una regione R_j sono classificate come l'evento più frequente, poiché minimizzano l'errore di classificazione empirico. L'obiettivo è trovare un albero in cui le osservazioni nei nodi terminali siano più omogenee possibili ed è quindi necessaria una misura di impurità. Il *mis-classification error* è semplicemente la frazione di osservazioni del training set che non appartengono alla regione con la più alta probabilità di appartenenza:

$$E = 1 - \max_k(\hat{p}_{jk})$$

dove $\hat{p}_{jk}$ è la proporzione delle osservazioni del training set nella *j*-esima regione e che provengono dalla *k*-esima classe. Esistono, però, misure di impurità migliori.

- L'indice di Gini:

$$G = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk})$$

Per un certo nodo *j,* dove $\hat{p}_{jk}(1 - \hat{p}_{jk})$ è la stima della varianza della distribuzione di Bernoulli: $Y_k^* = 1\, se\, Y = k$, la cui sommatoria fornisce la *varianza totale.* L'indice assume il valore zero se tutte le unità appartengono alla stessa categoria e un valore massimo se sono tutte in uguale proporzione.

- L'indice di entropia:

$$D = -\sum_{k=1}^{K} \hat{p}_{jk} \log \hat{p}_{jk}$$

Che è una misura di incertezza.

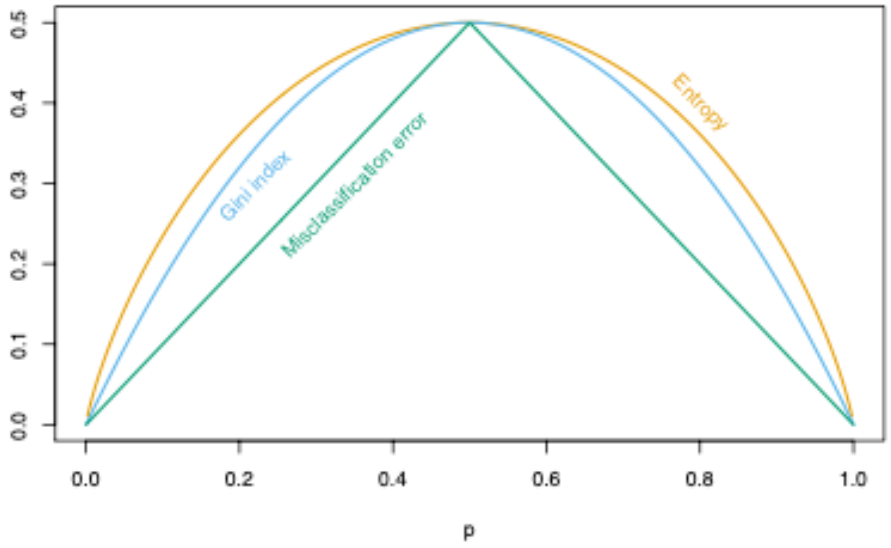

Figura 4: confronto tra misure di impurità di un nodo per una classificazione a due classi

Come è possibile notare dalla *figura 4,* sia l'indice di Gini che l'entropia sono comunemente usati per valutare la qualità di una specifica divisione e sono più sensibili alla *purezza* di un nodo rispetto al mis-classification error.

2.2 Vantaggi e svantaggi degli alberi

Il metodo degli alberi decisionali presenta dei vantaggi rispetto ai metodi classici:

1. Gli alberi sono di facile interpretazione. Questo è dovuto al pensiero che questa tecnica sia più vicina al processo decisionale umano rispetto ai metodi di classificazione affrontati in precedenza come la regressione logistica, LDA e il naive Bayes.
2. Gli alberi possono gestire facilmente e in modo naturale variabili qualitative senza dover creare variabili *dummy.*
3. È semplice computazionalmente e non è necessaria alcuna assunzione sulla distribuzione, essendo non-parametrica.

Gli alberi decisionali, presentano, però, anche una serie di svantaggi che, fortunatamente, possono essere risolti, almeno in parte, tramite particolari tecniche.

Un primo problema è la presenza di dati sbilanciati, complicazione di tutti i metodi di classificazione e in particolare negli alberi. In alcuni dataset può succedere che una classe sia molto più grande dell'altra e di conseguenza l'algoritmo cerca di minimizzare la misura di impurità, in particolar modo per la classe grande e perciò il predittore sarà distorto verso di essa. L'errore di classificazione sarà molto sbilanciato tra le classi e la performance sarà "gonfiata". Una possibile strategia e soluzione del problema potrebbe essere quella di bilanciare i dati campionando dalle osservazioni del training set. Nel caso di un campione grande è possibile usare solo una parte casuale della classe più numerosa, mentre, nel caso di un campione piccolo è possibile sovra campionare dalla classe meno numerosa.

Altri svantaggi, invece, sono dovuti ad alcune problematiche proprie degli alberi decisionali:

- Gli alberi, generalmente, non hanno lo stesso livello di accuratezza nelle previsioni delle altre tecniche di classificazione.
- Gli alberi non sono molto robusti. Un piccolo cambiamento nei dati può determinare un grande cambiamento delle divisioni binarie e quindi della struttura

dell'albero, rendendo la sua interpretabilità precaria.

Per fortuna, aggregando molti alberi, tramite le tecniche di *bagging, random forest* e *boosting,* la performance degli alberi può essere migliorata.

2.3 Bagging, Random Forests e Boosting

Gli alberi decisionali sono definiti *weak learners* [2] *poiché*, singolarmente, soffrono di un elevata varianza. Molti *weak learners* insieme, al contrario, possono ottenere stime migliori di un singolo algoritmo, grazie alla riduzione della varianza. Esistono varie tecniche di *combinazione* che uniscono tanti modelli "building block" per ottenere un singolo e potente modello.

2.3.1 Bagging

Il *bootstrap,* introdotto come strumento usato nei casi in cui è impossibile calcolare direttamente la deviazione standard della distribuzione di interesse, può essere utilizzato anche per migliorare tecniche di *statistical learning* come, in questo caso, gli alberi decisionali, i quali, hanno il "difetto" di avere una varianza elevata. Ciò significa che se le osservazioni del training set sono divise in due campioni casuali e su ognuno di essi costruisco un albero, quest'ultimi saranno molto diversi tra loro. Il *bagging* consiste proprio in una procedura di riduzione della varianza data dalla media ottenuta dagli alberi costruiti ognuno su un training set bootstrap diverso.

L'idea alla base è costruire una serie di alberi e ottenere da ognuno di essi una stima per poterne fare la media e ridurre, in questo modo, la variabilità della previsione. Formalmente:

$$X_1, \dots, X_n \sim_{i.i.d.} f(u, \sigma^2) \rightarrow \bar{X} \sim f^*(u, \frac{\sigma^2}{n})$$

Tuttavia, l'algoritmo per la costruzione di un albero, non essendo stocastico ma deterministico, produce sempre gli stessi risultati se utilizzato sugli stessi dati. È necessario introdurre un elemento casuale che è proprio dato dalla costruzione degli alberi su diversi campioni. Di conseguenza, un modo naturale per ottenere ciò è utilizzare tanti

training set B, costruire con essi altrettanti modelli $\hat{f}^1(x), \hat{f}^2(x), \dots, \hat{f}^B(x)$, fare la media delle stime e ottenere un singolo modello con varianza bassa:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x)$$

Il metodo in questione è corretto ma nella realtà è poco pratico in quanto è difficile avere un numero elevato di training set ed è per questo motivo necessario ricorrere al bootstrap. Comincio col bootstrap del training set D e ottengo B campioni bootstrap D_b. Per ogni campione bootstrap controllo quali osservazioni non sono incluse, creo un albero T_b senza "pruning" e calcolo la stima per ogni osservazione x: $\hat{f}^{*b}(x) = T_b(x)$. Infine, facendo la media delle stime ottengo:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Estendere tale metodo alla classificazione è relativamente semplice, *figura 5*. Per ogni osservazione utilizzata controlliamo la classe stimata da ciascun albero T_b e indichiamo come stima definitiva la classe più comune tra le previsioni degli alberi. Formalmente:

$$\hat{G}_{bag}(x) = argmax_k\, \hat{f}_{bag}(x), \qquad k \in \{1, \dots, k\}$$

dove K è il numero delle classi, $\hat{f}_{bag}(x)$ è un vettore $[p_1(x), p_2(x), \dots, p_K(x)]$ con $p_k(x)$ proporzione di alberi che stimano la classe k per x.

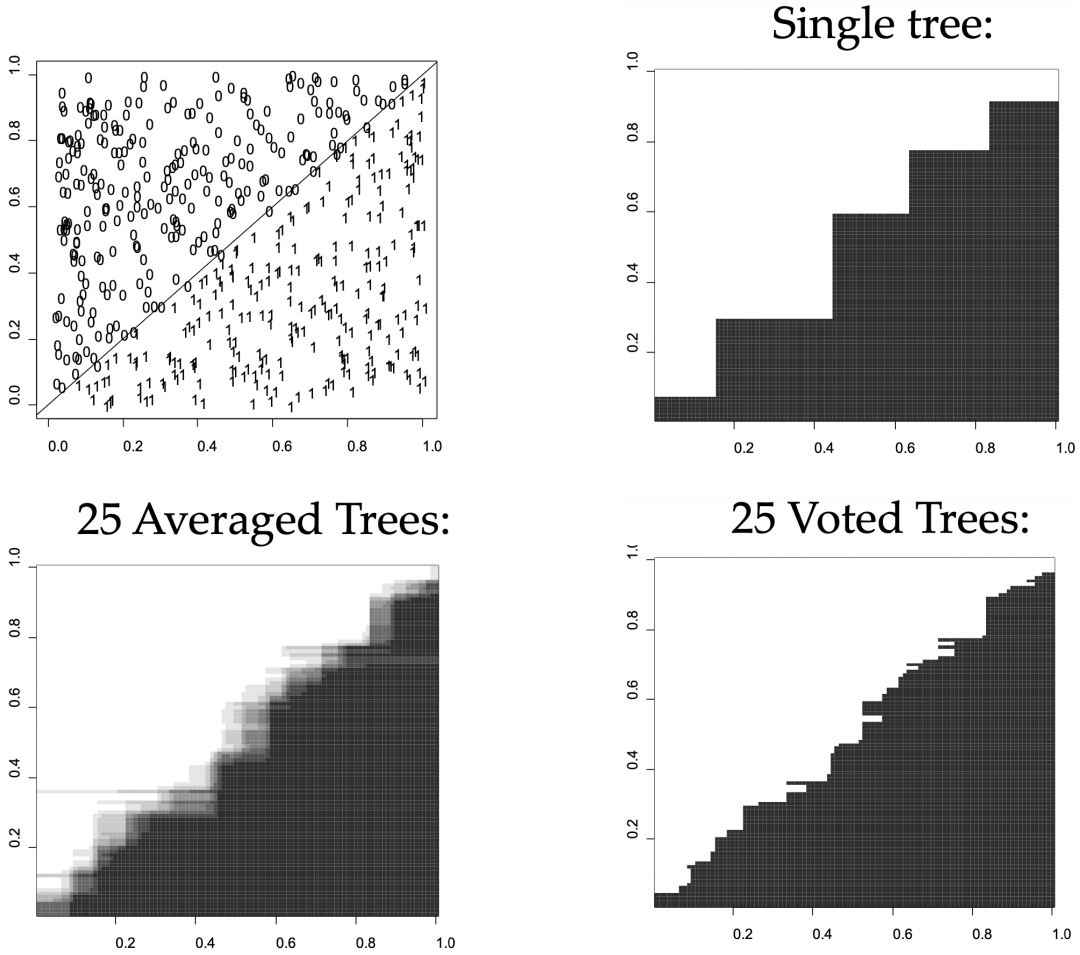

figura 5: processo di classificazione attraverso bagging

Esiste un metodo molto diretto per stimare l'errore di un modello di questo tipo. Ogni albero T_b, in media, usa due terzi delle osservazioni, lasciandone fuori un terzo, che definiamo osservazioni *out-of-bag.* Possiamo stimare la variabile di risposta di ogni osservazione utilizzando ciascun albero in cui essa non è presente e ottenendo, con questo metodo, circa $\frac{B}{3}$ stime per ogni osservazione. Come nel bagging normale la stima finale OOB sarà singola per ogni osservazione e con esse è possibile calcolare l'errore di classificazione. L'errore OOB è una stima valida poiché la risposta di ogni osservazione è stimata usando solo gli alberi che non erano allenati su quella osservazione.

2.3.2 Random forest

L'idea di base dietro la tecnica del bagging è fare la media di molti modelli *noisy* ma non distorti, al fine di ridurre la varianza, ma, siccome, ogni albero generato nel bagging è identicamente distribuito il *bias* dell'albero "bagged" è uguale a quello di ciascun albero bootstrap preso singolarmente. Una media di B $i.i.d.$ variabili omoschedastiche di varianza σ^2, ha varianza $\frac{\sigma^2}{B}$. Se le variabili sono solo $i.d.$, come in questo caso, con correlazione positiva a coppie pari a ρ, la varianza della media è:

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Al crescere di B il secondo termine sparisce, ma il primo, in cui è presente il termine di correlazione tra coppie, rimane. L'obiettivo delle *random forest* è quello di ridurre la correlazione tra gli alberi, senza aumentare la varianza [2]. Questo è possibili durante il processo di costruzione dell'albero con la scelta casuale delle variabili. Quando costruiamo un albero su un campione bootstrap, prima di ogni divisione, selezioniamo un numero $m \leq p$ delle variabili come plausibili candidati per la divisione. Tipicamente il valore di m è $\sqrt{p}$. l'algoritmo dietro le random forest [2] è il seguente:

1. Per b da 1 a B:
    a. Crea un campione bootstrap Z^* di grandezza N dalle osservazioni del training set.
    b. Costruisci un albero random forest T_b dai dati del campione, ripetendo ricorsivamente i seguenti passi per ogni nodo terminale dell'albero, finché non è raggiunta la grandezza minima del nodo n_{min}:
        I. Seleziona m variabili casualmente da p variabili;
        II. Scegli la miglior variabili e punto di divisione tra m
        III. Dividi il nodo in due nodi figli.
2. Mostra l'insieme di alberi $\{T_b\}_1^B$.

Per fare stime dato un valore x:

- $\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$ *regressione;*
- $\hat{C}_{rf}^{B}(x) = argmax\{\hat{C}_b(x)\}_1^B$ *classificazione.*

Dove $\hat{C}_b(x)$ è lo stimatore di classe del b-esimo albero random forest.

In alter parole, nel processo di costruzione di una random forest, ad ogni divisione dell'albero, l'algoritmo non è nemmeno "autorizzato" a considerare una maggioranza delle possibili variabili. Potrebbe sembrare strano ma c'è una motivazione ben precisa. Nel caso in cui, nel dataset, ci fosse una variabile predominante sulle altre, nell'insieme di alberi del bagging la maggior parte, se non tutti, userebbero questa variabile alla prima divisione. Di conseguenza tutti gli alberi sarebbero simili tra loro e pertanto le stime avrebbero un elevata correlazione. Possiamo considerare l'algoritmo come un processo di de-correlazione degli alberi che rende la media degli alberi meno variabile e quindi più affidabile.

2.3.3 Misure di importanza di una variabile

I metodi del bagging e delle random forest portano, tipicamente, ad una stima migliore rispetto all'utilizzo di un solo albero. Tuttavia, può essere difficile interpretare il risultato del modello. Quando uniamo un numero elevato di alberi, non è più chiaro quali siano le variabili più importanti e quindi avremo una perdita dell'interpretabilità al fine di ottenere una stima precisa. Sebbene l'insieme di alberi random forest sia più difficile da interpretare di un singolo albero, è possibile ottenere un indice di importanza generale per ciascuna variabile usando l'indice di Gini [1]. È possibile sommare quanto l'indice di Gini diminuisce dopo la divisione dovuta a una data variabile e fare la media per ogni albero B. Un altro metodo per misurare l'importanza della variabile si basa sull'idea che se una variabile è irrilevante, allora errore di stima dovrà essere diverso quando essa è permutata. Questa tecnica utilizza i campioni OOB che vengono passati per il *b*-esimo albero e si registra la precisione. Poi i valori per la *j*-esima variabile sono permutati casualmente nei campioni OOB, e si calcola nuovamente la precisione. La decrescita di precisione come risultato di questa permutazione è mediata su tutti gli alberi ed è usata come misura di importanza della variabile *j* nelle random forest.

2.3.4 Boosting

Il *boosting* è uno dei più potenti strumenti introdotto negli ultimi 20 anni. Creato, inizialmente, proprio per problemi di classificazioni ma usato anche per la regressione. Il boosting è sostanzialmente un metodo che combina l'*output* di tanti "weak classifiers" [1] per produrre uno "strong classifier". L'idea alla base è simile a tecniche come il bagging o le random forest ma c'è una sostanziale differenza: come si vede dalla *figura 6*, mentre il bagging e le random forest creano alberi in *parallelo, il boosting* lo fa *sequenzialmente*, costruendo un albero che prova a "compensare" per gli errori degli alberi precedenti.

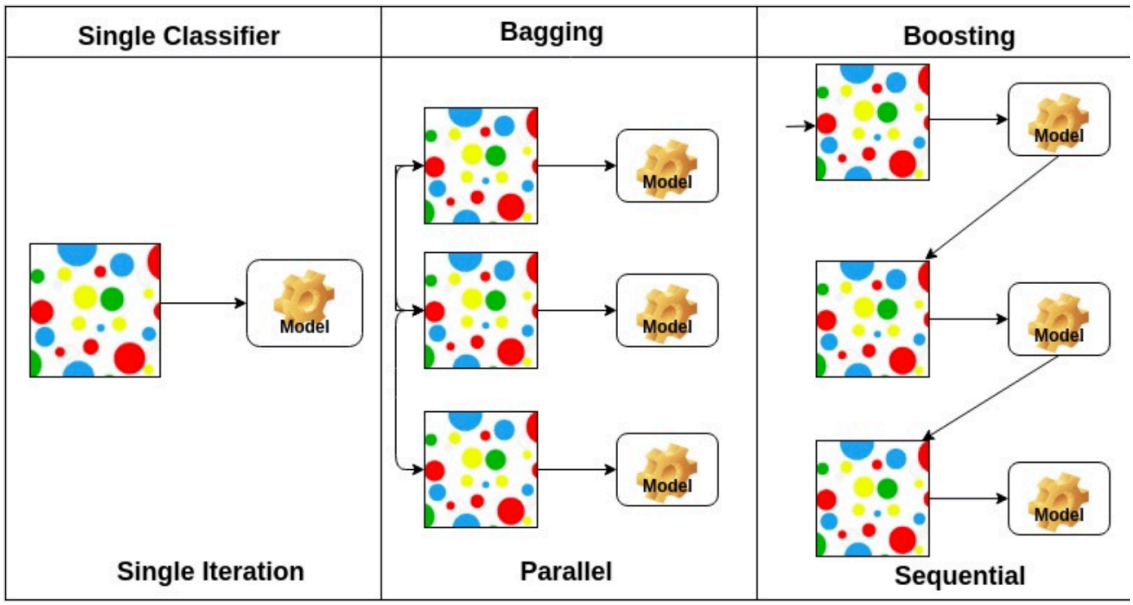

Figura 6: confronto tra struttura parallela del bagging e sequenziale del boosting

Esistono tanti algoritmi di boosting ma il più popolare è sicuramente l'*AdaBoost,* ideato da Freund e Schapire. Considerato un problema a due classi, con le variabili di risposta $Y \in \{-1,1\}$ e dato un vettore di variabili aleatorie X, un classificatore $G(X)$ esegue una stima prendendo uno dei due valori $\{-1,1\}$. Il tasso di errore sul campione di addestramento è:

$$\overline{err} = \frac{1}{N}\sum_{i=1}^{N} I(y_i = G(x_i))$$

e il tasso di errore atteso sulle stime future è $E_{XY}I\big(Y \neq G(X)\big)$. Un classificatore è debole quando il suo tasso di errore è solo poco meglio di stimare a caso. L'obiettivo del boosting è di applicare *sequenzialmente* l'algoritmo di classificazione debole a versioni dei dati ripetutamente modificate, in modo da avere una sequenza di classificatori deboli $G_m(x), m = 1,2, \dots, M$. Le stima di tutti questi classificatori sono poi combinate tramite un voto a maggioranza ponderata per ottenere la stima finale:

$$G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

dove $\alpha_1, \alpha_2, \dots, \alpha_M$ sono parametri calcolati dall'algoritmo per assegnare il peso a ogni rispettivo classificatore debole.

Il boosting può essere applicato a tanti metodi di apprendimento statistico ma si applica particolarmente bene con gli alberi. Ogni albero T_b è creato usando le informazioni dell'albero precedente. Non è utilizzato un campionamento bootstrap ma ogni albero è allenato su una versione modificata del set di dati originali. Nella regressione il boosting combina un grande numero di alberi $\hat{f}^1, \dots, \hat{f}^B$ ma, invece di utilizzare un singolo grande albero decisionali, impara "lentamente". Dato il modello corrente, adattiamo l'albero decisionale ai residui del modello, invece della risposta *Y* e poi aggiungiamo il nuovo albero decisionale nella funzione per aggiornare i residui. Così facendo miglioriamo lentamente $\hat{f}$. formalmente [2] l'algoritmo applicato agli alberi di regressione è il seguente:

1. Set $\hat{f}(x) = 0$ *e* $r_i = y_i$ per ogni *i* nel training set.
2. For $b = 1,2, \dots, B$, esegui i seguenti passaggi:
    a. Adatta un albero $\hat{f}^b$ con d divisioni alle osservazioni del training set (X, r)
    b. Aggiorna $\hat{f}$ aggiungendo in una versione ridotta del nuovo albero:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

c. Aggiorna i residui:

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

3. Calcola il modello finale:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Il boosting ha tre parametri regolatori:

I. Il numero di alberi B. A differenza del bagging o delle random forest, il boosting può avere *overfitting* se B è troppo grande. Si usa la *cross-validation* per selezionare B.

II. Il parametro *shrinkage* λ, un piccolo numero positivo che controlla a quale "velocità" l'algoritmo impara. Valori tipici sono 0.01 e 0.001.

III. Il numero d di divisioni in ogni albero, che controlla la complessità del boosting. Spesso $d = 1$ funziona bene. In generale d è l'*interaction depth* e controlla l'ordine di interazione del modello boosting, visto che d divisioni implicano massimo d variabili.

Il boosting per gli alberi di classificazioni procede nello stesso modo.

I metodi basati su alberi sono strumenti molto potenti per la classificazione. Dopo aver visto come far crescere un albero decisionale, si è reso necessario, per risolvere alcune delle problematiche legati all'uso di un singolo albero, come l'instabilità e una alta varianza, l'utilizzo di alcuni *ensemble methods*, tra cui il bagging, le random forest e il boosting. Gli alberi sono, infatti, un'ottima scelta di *weak learners* per la loro flessibilità e capacità di gestire variabili quantitative e qualitative allo stesso tempo. Ogni ensemble methods ha caratteristiche diverse [1]:

- Nel *bagging,* gli alberi crescono indipendentemente su osservazioni da campioni casuali e di conseguenza, tendono ad essere simili tra loro. Per questo motivo il bagging può rimanere "incastrato" sugli ottimi locali e non riuscire a esplorare a fondo lo spazio del modello.

- Nelle *Random forests*, gli alberi crescono indipendentemente su osservazioni da campioni casuali ma, ogni divisione su ogni albero è scelta da un sotto campione casuale delle variabili, in modo da de-correlare gli alberi e avere un'esplorazione migliore dello spazio del modello.
- Nel *boosting,* sono usati solo i dati originali. gli alberi crescono in maniera sequenziale con un apprendimento "lento": ogni nuovo albero è addestrato sui residui dell'albero precedente e rimpicciolito prima di usarlo.

3. Applicazioni empiriche dei metodi basati su alberi

Dopo aver analizzato vari aspetti dei metodi di classificazione parametrici tra cui la regressione logistica, l'analisi discriminante lineare e il naive Bayes ed i punti di forza di ciascuno, l'attenzione si è spostata verso un particolare metodo di classificazione non-parametrico: gli alberi decisionali. In questo capitolo, con il supporto teorico descritto precedentemente, l'obiettivo è svolgere analisi empiriche su dati del mondo reale, confrontare le tecniche basate sugli alberi, dalle più facili alle più complicate, e trarre conclusioni su quale si presta meglio in base a molteplici situazioni e ambiti di applicazione.

3.1 Classificazione dei tumori tramite alberi decisionali

Le tecniche di classificazione hanno molti campi di applicazione, tra cui quello medico. In questo caso, gli alberi decisionali sono sfruttati per la classificazione della pericolosità del tumore al seno. Il dataset [5] è composto da 569 osservazioni, di cui 357 benigni e 212 maligni, e da dieci variabili casuali principali che forniscono misure riguardo forma e dimensioni di esso. In particolare, le variabili sono: il raggio, la consistenza, il perimetro, l'area, levigatezza, compattezza, concavità, numero di punti concavi, simmetria, dimensione frattale. Di ognuna di esse sono state rilevate la media, la deviazione standard e la media dei tre valori più grandi, per un totale di trenta variabili casuali. L'obiettivo è confrontare la precisione dei metodi di classificazione basati su alberi che sono stati analizzati nel secondo capitolo.

Inizialmente, un semplice albero di classificazione è applicato a tutti i dati, ottenendo un tasso di errore di classificazione di solo 1.7%. Gli alberi sono facilmente interpretativi dal punto di vista grafico, è quindi possibile visualizzare l'albero, *figura 7,* in questione che fornisce già un'idea dell'importanza delle variabili; in particolar modo sembra che la variabile più importante sia "perimeter worst". Allo stesso tempo è subito visibile un numero troppo elevato di nodi che dovrà essere ridotto tramite il *pruning.*

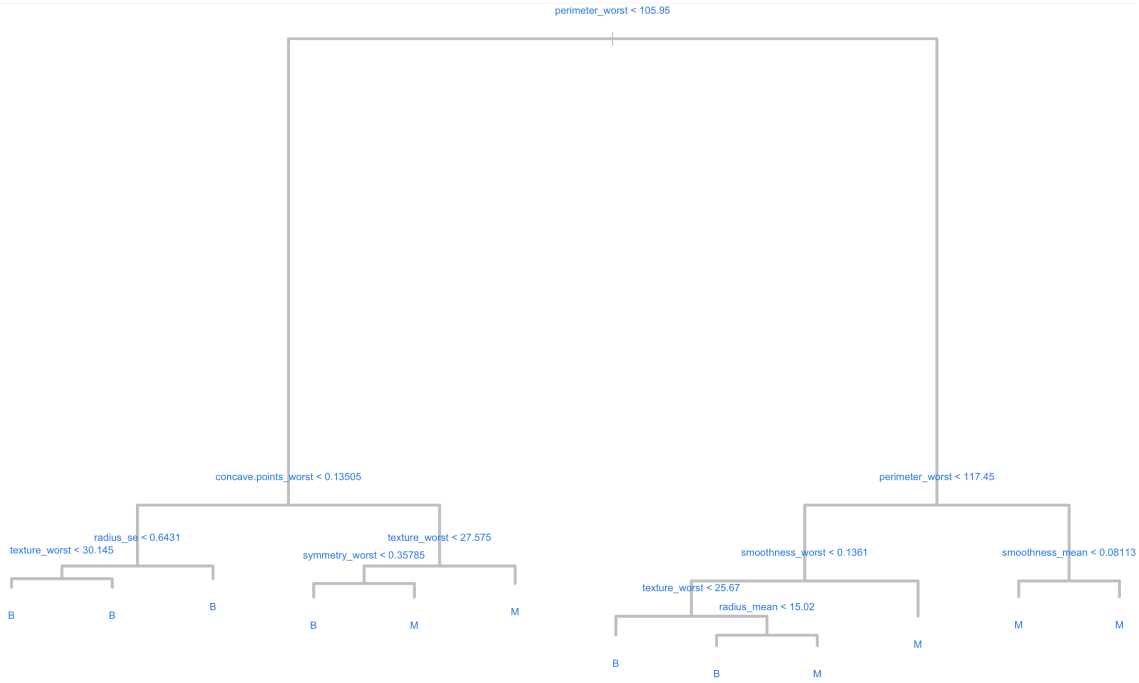

figura 7: Albero di classificazione

Per valutare in maniera appropriata la performance di un albero di classificazione, però, è necessario stimare il *test error* e non limitarsi al *training error*. Le osservazioni sono divise in un *training set* sul quale sarà addestrato l'albero e in un *test set* sul quale sarà valutato. Il modello è addestrato su 250 osservazioni e le restanti 319 sono utilizzate per testarlo. Ciò che ne risulta, dalla matrice di confusione è un'accuratezza del 94,36%. 197 osservazioni sono correttamente classificate come benigne e solo 7 sono state incorrettamente classificate come maligne. Al contrario, 108 osservazioni sono state classificate correttamente come maligne e solo 11 incorrettamente come benigne.

	B	M
B	193	7
M	11	108

Tramite la tecnica del *pruning* la performance può essere migliorata in quanto risolve il problema dell'overfitting. È fondamentale la *cross-validation* per determinare il livello ottimo di complessità. Come è possibile vedere in *figura 8,* l'albero con quattro nodi terminali è quello con meno errori di cross-validation pari a 21 e $\alpha = 1$.

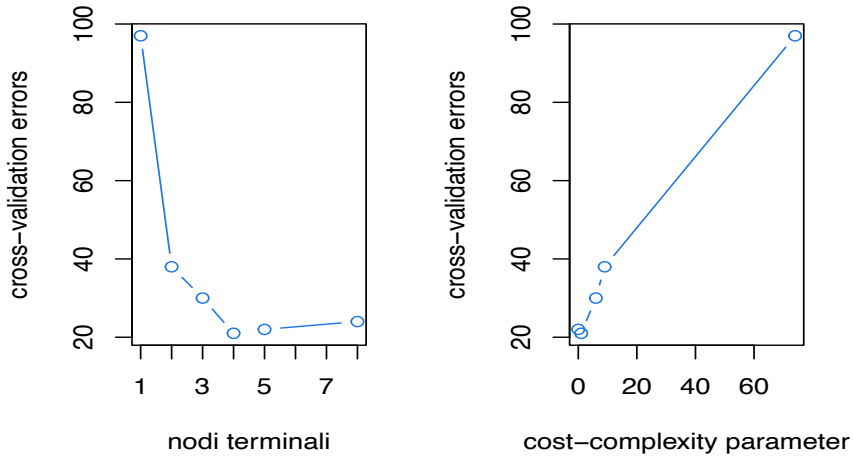

Figura 8: grafico del numero di errori della cross-validation in base al numero di nodi terminali e al parametro di costo di complessità

Una volta applicata la tecnica del pruning e ottenuto l'albero con quattro nodi terminali è possibile visualizzarlo graficamente, *figura 9*.

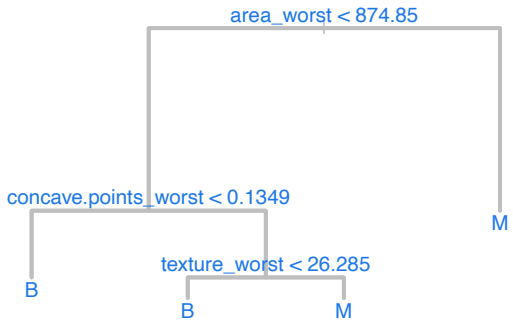

Figura 9: albero di classificazione dopo il pruning

L'albero è molto più interpretabile rispetto a quello iniziale con dodici nodi terminali.

Inoltre, se applichiamo il test set all'albero ottenuto e analizziamo la matrice di confusione è evidente anche un miglioramento delle performance e un'accuratezza pari al 95,3%.

	B	M
B	196	7
M	8	108

Gli alberi decisionali hanno il difetto di avere una varianza elevata, che può essere ridotta tramite il *bagging* che permette di costruire un numero finito di alberi su campionamenti diversi, ottenuti tramite *bootstrap,* e farne la media. Per questo motivo il bagging, con 500 alberi, è applicato al training set ed è possibile ottenere gli errori di classificazione, *figura 10*, in particolare l'errore delle stime *out-of-bag* è il 7.2%.

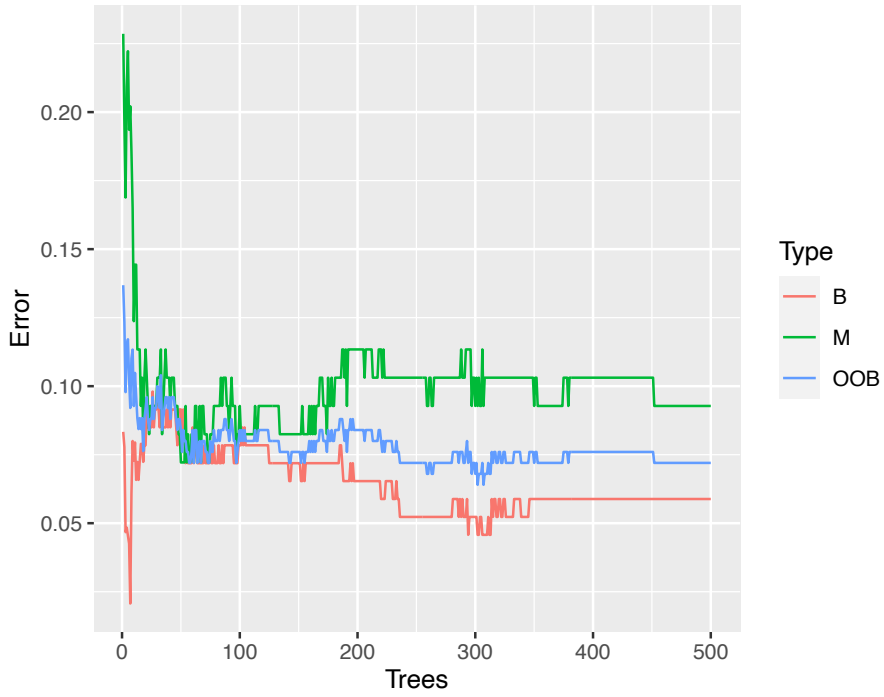

Figura 10: errori di classificazione delle osservazioni benigne, maligne e out-of-bag

Applicando il modello al test set si ottiene un ulteriore incremento dell'accuratezza, adesso pari al 96.24%.

Attraverso una semplice analisi grafica, *figura 11*, raffigurante una parte degli *scatterplot* tra variabili è possibile denotare la presenza o meno di correlazione tra coppie che, nel caso sia elevata può limitare i benefici del bagging.

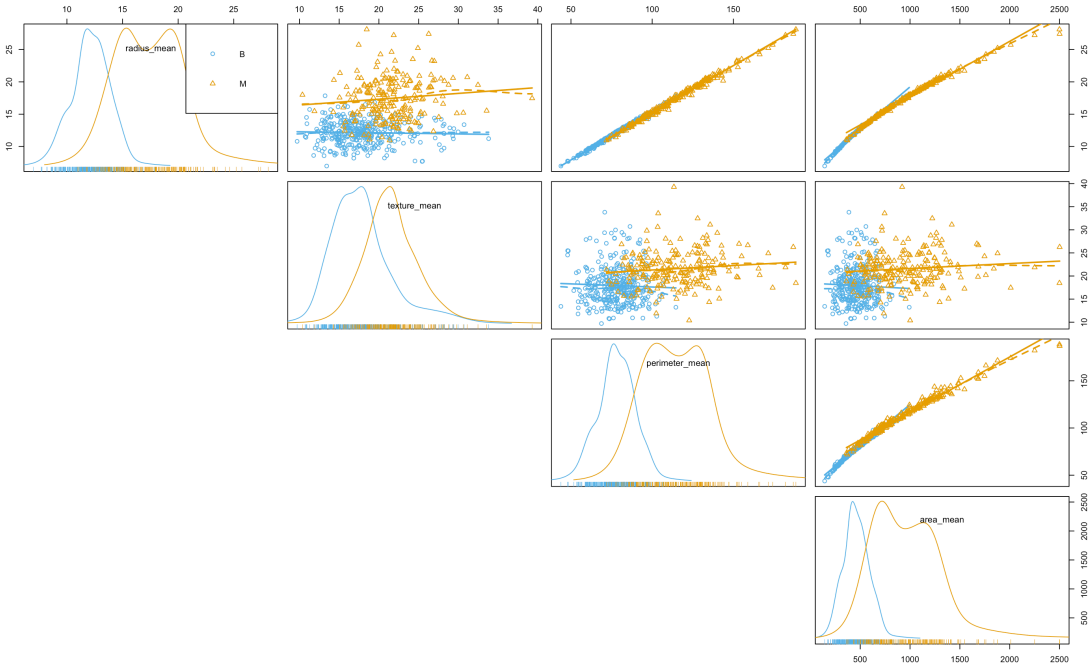

Figura 11: matrice scatterplot delle per la correlazione tra variabili

Tra molte variabili la correlazione è molto elevata. Per risolvere questo problema è possibile utilizzare la tecnica delle *random forest* e quindi far in modo che solo un sottogruppo delle variabili sia disponibile per decidere il miglior criterio di divisione in ogni campione bootstrap. Il numero di variabili tipico è pari alla radice quadrata del numero totale di variabili e quindi, in questo caso, approssimando per difetto, è 5. Allenando il modello sul solito training set si ottiene una stima dell'errore di classificazione sulle osservazioni out-of-bag pari al 5.2%, *figura 12,* mentre la matrice di confusione restituisce errori di classificazione pari al 3.9% e al 7.2% rispettivamente per le osservazioni benigne e maligne. L'errore è minore rispetto a quello del bagging e, per questo motivo, è naturale aspettarsi un miglioramento sulla classificazione del test set.

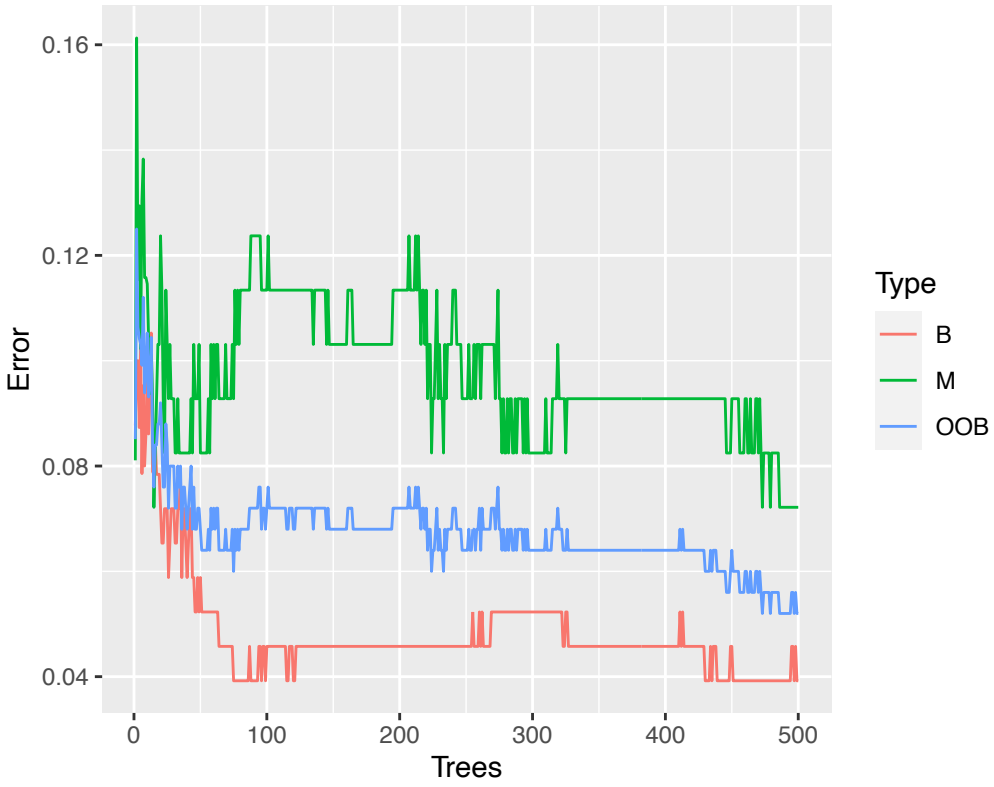

figura 12: errori di classificazione delle osservazioni benigne, maligne e out-of-bag

Applicando il modello al test set, si ottiene un'accuratezza pari al 96,87% e la matrice di confusione seguente:

	B	M
B	201	7
M	3	108

A differenza del primo albero decisionale che poteva essere visto graficamente, adesso non è più possibile e per questo si ha con le random forest una perdita dell'interpretabilità che non permette più di capire facilmente quali siano le variabili più rilevanti. Per fortuna esistono misure per calcolare l'importanza delle variabili. La prima si basa sui campioni OOB mentre la seconda consiste nel sommare quanto diminuisce l'indice di Gini dopo la divisione dovuta a una data variabile e fare la media per ogni albero.

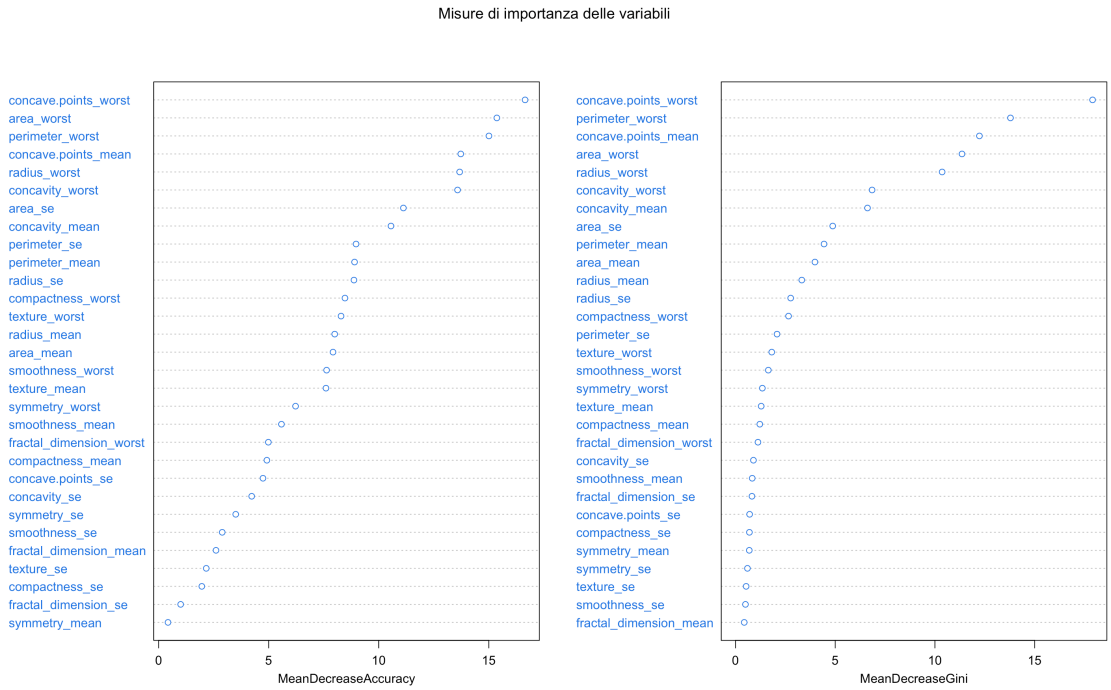

figura 13: misure di importanza delle variabili

In particolar modo, le variabili più influenti, *figura 13,* sono: “concave points worst”, “area worst”, “perimeter worst”, “concave points mean” e “radius worst”, che occupano le prime cinque posizioni in entrambi gli indici. Le variabili meno importanti, invece, sono: “symmetry mean” e “fractal dimension se” per la *mean decrease accuracy* e “fractal dimension mean” e “smoothness se” per la *mean decrease Gini.*

L’ultimo modello basato su alberi da valutare è il *boosting* che si differenzia dai precedenti per la crescita sequenziale degli alberi, invece che parallelamente. Il boosting è applicato al training set che restituisce un grafico dell’influenza relativa delle variabili. Come nelle random forest, anche adesso le variabili più importanti, *figura 14,* risultano “concave points worst” e “area worst”, mentre le meno importanti sono “fractal dimension worst” e “area mean”.

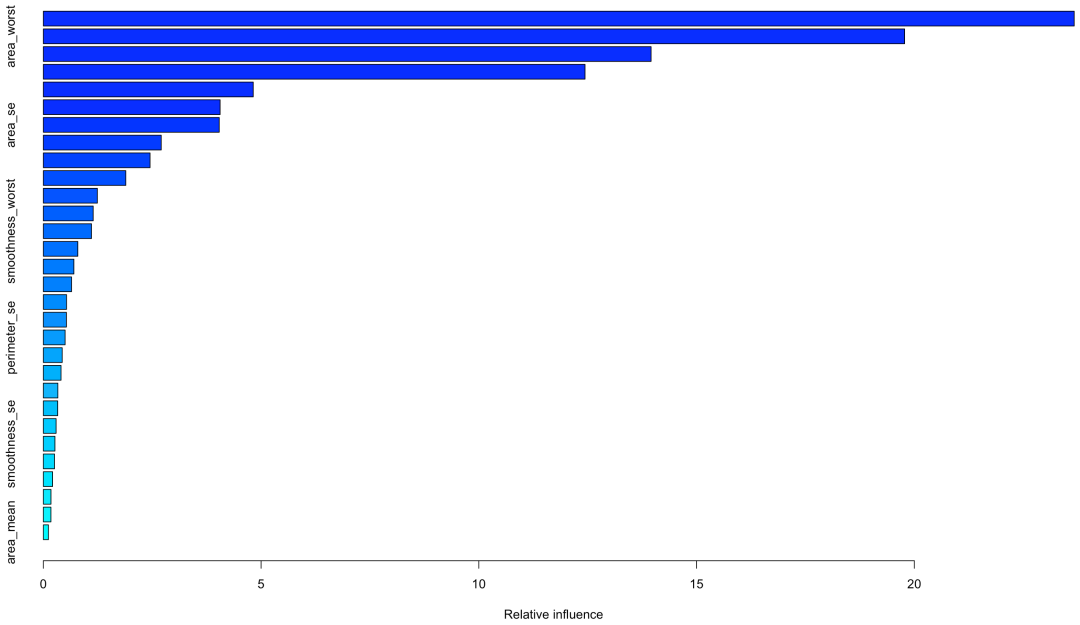

Figura 14: influenza relativa

Applicando il modello addestrato sul test set si ottiene la percentuale di accuratezza migliore finora, pari al 97,18% e la seguente matrice di confusione:

	B	M
B	199	4
M	5	111

Il box plot in *figura 15* mette a confronto le performance dei metodi basati sugli alberi che sono stati applicati al dataset. Sono riportati anche i relativi intervalli di confidenza al 95% ed è quindi evidente, anche dal punto di vista grafico, che il metodo del boosting sia il più appropriato per la classificazione dei tumori al seno.

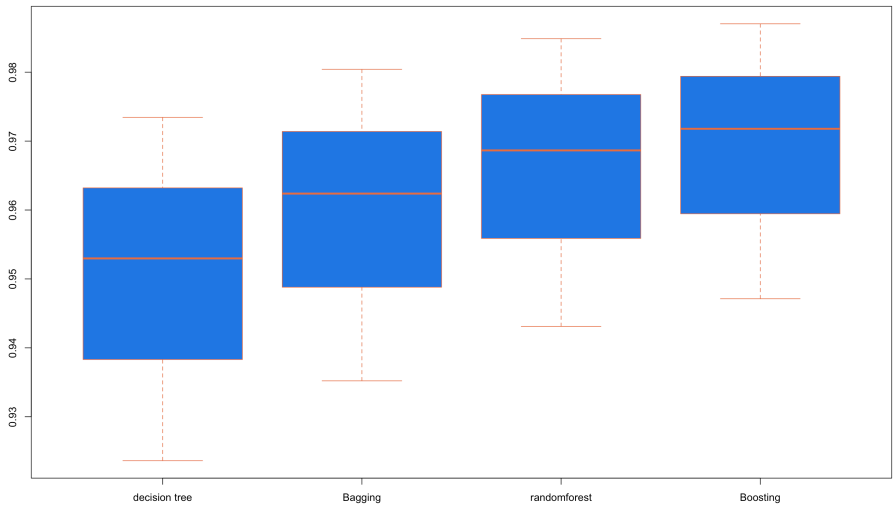

Figura 15: Boxplot per il confronto dell'accuratezza

	Accuratezza	Sensibilità	Specificità	Precisione	F1
DecisionTree	**0.9529781**	**0.9608**	**0.9391**	**0.9655**	**0.9631**
Bagging	**0.9623824**	**0.9657**	**0.9565**	**0.9752**	**0.9704**
RandomForest	**0.9686520**	**0.9853**	**0.9391**	**0.9663**	**0.9757**
Boosting	**0.9717868**	**0.9755**	**0.9652**	**0.9803**	**0.9779**

Tabella 1: misure di confronto tra i metodi

3.2 Modelli parametrici e non parametrici per la qualità del vino

I modelli di classificazione statistica si dividono in due grandi tipologie: modelli parametrici e non parametrici [8]. Non esiste a priori un metodo migliore ma, la scelta di quale applicare, dipende dalle singole situazioni e dalla tipologia di dati sui quali lavorare. I modelli parametrici si basano su specifiche ipotesi riguardo la relazione dei dati in entrata ed in uscita. Queste assunzioni sono un numero fissato di parametri e l'obbligo che la popolazione sia distribuita come una normale, o che si possa approssimare ad essa. Questi modelli hanno, in base a queste ipotesi, vantaggi e svantaggi. Non hanno bisogno di molti dati per raggiungere un certo livello di precisione e sono computazionalmente efficienti poiché imparano un numero fisso di variabili basate sulle ipotesi. Dall'altra parte, spesso, le ipotesi semplificano troppo il problema e perciò il modello non è in grado di capire comportamenti complessi e relazioni tra i dati. Infine, i modelli parametrici sono sensibili agli *outliers* e la loro performance è limitata per problemi non lineari. I modelli non parametrici non devono fare ipotesi riguardo la relazione tra dati in entrata ed in uscita e non richiedono un numero fisso di parametri da individuare e imparare. Inoltre, questi modelli funzionano meglio su grandi moli di dati e sono più flessibili. Sono, quindi, in grado di catturare comportamenti complessi e relazioni tra i dati senza seguire delle ipotesi. Possono gestire efficacemente gli outliers e dati non lineari in quanto sono flessibili e adattabili in base alla situazione. Gli svantaggi principali di questi modelli sono la necessità di molti dati per poter e generare stime efficienti e sono computazionalmente dispendiosi in quanto devono stimare più parametri.

L'obiettivo di questa analisi è confrontare l'efficacia di classificazione di modelli parametrici, quali: la regressione logistica, l'analisi discriminante lineare e il naive Bayes con i modelli non parametrici basati sugli alberi decisionali. Il dataset in questione [6] è composto da 1599 osservazioni e 11 variabili casuali quantitative: "fixed acidity", "volatile acidity", "citric acidity", "residual sugar", "chlorides", "free sulfur dioxide", "total sulfur dioxide", "density", "pH", "sulphates", alcohol" e la variabile di risposta "good" che indica se un vino sia buono o meno. Come prima cosa l'analisi visiva degli scatterplot e della distribuzione delle di alcune delle variabili fornisce un'idea su quali tipologia di modelli utilizzare.

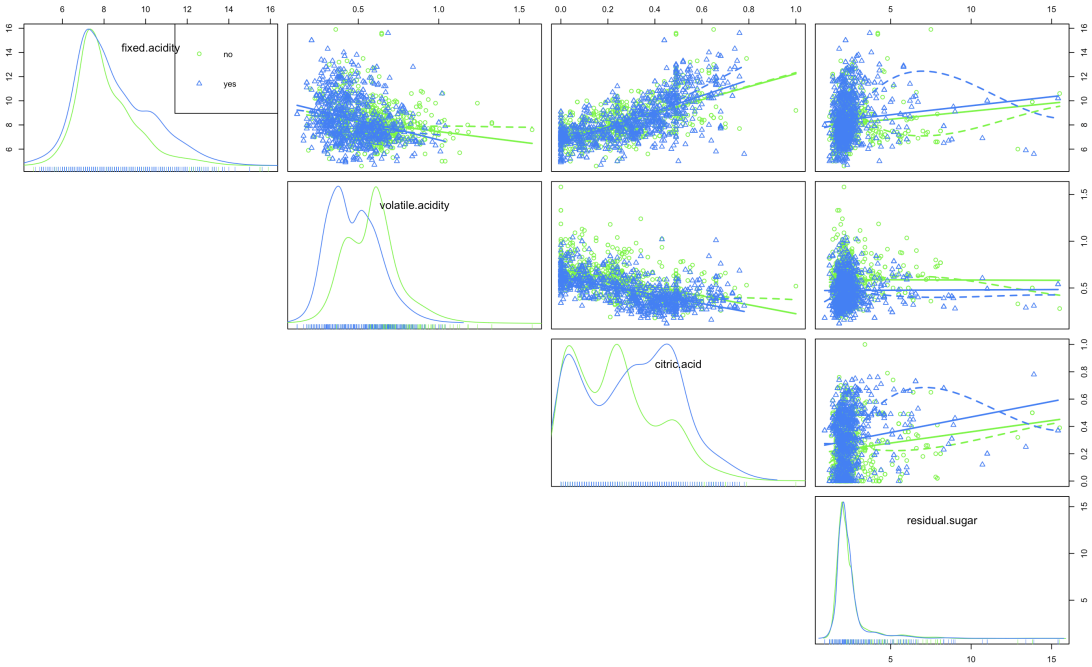

Figura 16: matrice scatterplot delle per la correlazione tra variabili

Le variabili non hanno una distribuzione normale, *figura 16*, verificabile anche con il test di Shapiro, e ciò può far intuire che l'utilizzo di modelli basati su alberi possa risultare più adatto rispetto ai modelli parametrici.

il dataset è diviso in training set, composto da 1000 osservazioni, e test set. Il primo modello sperimentato è la regressione logistica che, dopo aver rimosso le variabili non significative e quindi inutili per la stima, la precisione di stima che ne risulta è pari al 74.1% e la matrice di confusione è la seguente:

	No	Yes
No	213	89
Yes	66	231

Il secondo modello è l'analisi discriminante lineare. Le probabilità a priori sono $\hat{\pi}_1 = 0.465$ *e* $\hat{\pi}_2 = 0.535$, cioè il 46.5% delle osservazioni del training set corrisponde ai vini non considerati buoni e il 53.5% sono considerati buoni.

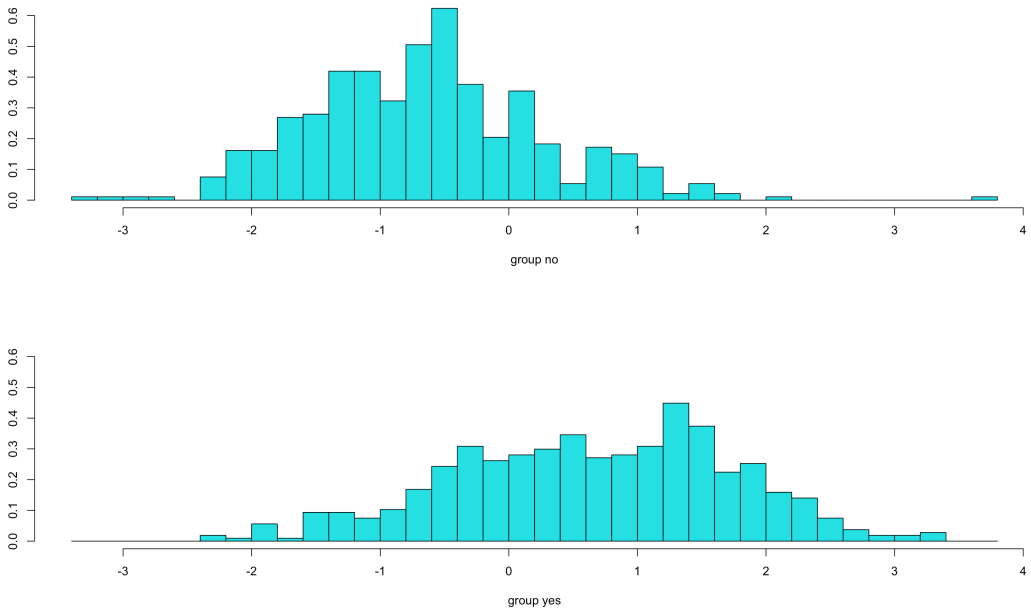

Figura 17: istogramma LD1

Il grafico in *figura 17* mostra una notevole sovrapposizione tra le osservazioni dei due gruppi e per questo probabilmente le nostre stime non potranno essere molto accurate. Infatti, la precisione di stima è addirittura più bassa della regressione logistica, pari al 72.95% e la matrice di confusione è la seguente:

	No	Yes
No	211	94
Yes	68	226

L'ultimo modello parametrico è il naive Bayes che, rispetto all'analisi discriminante lineare migliora leggermente la precisione di stima, avvicinandosi a quella della regressione logistica. La precisione di stima è 73.96% e la matrice di confusione è la seguente:

	No	Yes
No	204	81
Yes	75	239

I modelli basati su alberi forniscono stime migliori su questo dataset, sia grazie all'elevato numero di osservazioni disponibili per training set che per la distribuzione dei dati complessa. Il bagging ha una precisione pari al 79.8% che è poco inferiore a quella delle random forest: 80.63%. un miglioramento così basso è dovuto alla già bassa correlazione a coppie di variabili che quindi non è ulteriormente controllata con la tecnica delle random forest. Il Boosting ha la precisione di stima più bassa tra i tre, pari al 77.8%. la matrice di confusione della random forest è la seguente:

	No	Yes
No	229	66
Yes	50	254

il boxplot in *figura 18* confronta i metodi di classificazione parametrici e la classificazione random forest in quanto è stata la migliore tra quelli non parametrici ed è evidente, pure visivamente, che in questo caso sia stato favorevole l'utilizzo di una tecnica non parametrica.

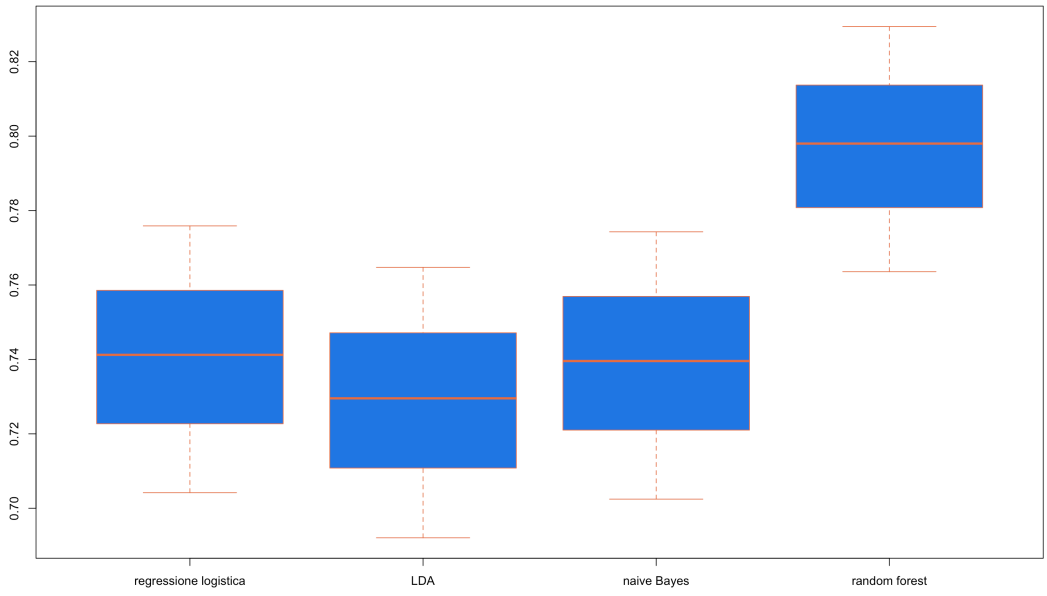

figura 18: Boxplot di confronto tra i metodi

	Accuratezza	Sensibilità	Specificità	Precisione	F1
RegLogistica	**0.7412**	**0.7634**	**0.7219**	**0.7053**	**0.7229**
LDA	**0.7295**	**0.7563**	**0.7063**	**0.6918**	**0.7102**
NaiveBayes	**0.7396**	**0.7312**	**0.7469**	**0.7158**	**0.7234**
RandomForest	**0.798**	**0.8100**	**0.7875**	**0.7687**	**0.7889**

Tabella 2: misure di confronto tra i metodi

Conclusione

In una realtà sempre più *data-driven* l'utilizzo di modelli statistici è indispensabile per stare al passo con i tempi. Ciò che rende questi strumenti indispensabili è la loro eterogeneità di applicazione. Ogni settore: medico, economico, sociale, ingegneristico può trarne benefico se usati correttamente. Lo scopo del presente elaborato è sfruttare questi modelli per mostrare il loro funzionamento, mostrare i punti di forza e debolezza e tipici utilizzi pratici. Inizialmente sono stati esposti i modelli di classificazione più comuni e semplici, in particolar modo la regressione logistica che rimane comunque uno degli strumenti più utilizzati. Altri metodi che appartengono sempre alla stessa famiglia dei modelli parametrici sono l'analisi discriminante lineare e il naive Bayes che trovano il loro punto di forza nei casi in cui le classi siano più di due, riportando, in generale, risultati migliori rispetto alla regressione logistica. Fondamentale è stata anche l'introduzione di tecniche di ri-campionamento che sono, come abbiamo visto, strettamente connesse ai metodi di classificazione in quanto sono indispensabili per ottenere un buon risultato di stima, in particolar modo ai modelli non parametrici. Quest'ultimi si differenziano per la mancanza di parametri fissi che permettono una grande flessibilità ma, allo stesso tempo, se non si basano su tecniche come bootstrap e cross-validation possono portare a risultati ancora più distorti. Nel nostro caso l'attenzione è ricaduta sui metodi basati su alberi. Siamo partiti dal semplice albero decisionale, la sua costruzione e i suoi punti di vantaggio e svantaggi, per arrivare a più complicate tecniche di ensambling come il bagging, le random forest e il boosting. Di ogni algoritmo è stato spiegato il funzionamento, le similarità e le differenze tra essi. Dopo la teoria è stato anche fondamentale dare uno sguardo alla pratica, mettendo in funzione le tecniche analizzate. Abbiamo applicato i metodi basati su alberi a un dataset

per classificare i tumori al seno come benigni o maligni a seconda di dati quantitativi su di essi. Attraverso l'utilizzo del software R è stato possibili individuare la tecnica che ha portato al minor errore di classificazione che si è rivelato essere, in questo caso, il boosting. Una seconda analisi empirica ha permesso di confrontare su dati reali le tecniche parametriche e non esaminate. L'obiettivo era quindi quello di capire se in una certa situazione, a seconda della quantità di dati disponibili e della distribuzione di essi quale metodo fosse più efficace. Alla fine, è stato l'algoritmo random forest ad avere un minor errore di classificazione, migliorando la precisione di stima delle tecniche parametriche di circa il 7%. In un mondo in cui le disponibilità di dati sono sempre più elevati e spesso sono molto complessi, l'utilizzo di tecniche di classificazione non parametriche che, oltre agli alberi comprendono pure il *k-Nearest Neighbors* e il *support vector machines,* è fondamentale per molte grandi aziende che ormai non possono più farne a meno.

Appendice

```
library(caret)
library(tree)
library(ISLR2)
library(ggplot2)
library(cowplot)
data_cancer=read.csv2("/Users/leonardomichi/Desktop/data_cancer.csv",
header=T,sep=",",dec=".")
data_cancer=data.frame(data_cancer)
data_cancer=data_cancer[-c(1,33)]
data_cancer$diagnosis=factor(data_cancer$diagnosis)
set.seed(3)
#decision tree
tree.data_cancer=tree(diagnosis~.,data_cancer)
summary(tree.data_cancer)
plot(tree.data_cancer,col="grey",lwd=3,label_context)
text(tree.data_cancer,pretty=1,col="#1E77E8",cex=0.7)
tree.data_cancer
train=sample(1:nrow(data_cancer),250)
data_cancer.test=data_cancer[-train,]
diagnosis.test=data_cancer$diagnosis[-train]
tree.data_cancer=tree(data_cancer$diagnosis~.,data_cancer,subset=train)
tree.pred=predict(tree.data_cancer,data_cancer.test,type = "class")
#table(tree.pred,diagnosis.test)
confusionMatrix(tree.pred,diagnosis.test)
cv.data_cancer=cv.tree(tree.data_cancer,FUN=prune.misclass)
names(cv.data_cancer)
cv.data_cancer
par(mfrow=c(1,2))
plot(cv.data_cancer$size,cv.data_cancer$dev,type="b",col="#1E77E8",xlab="nodi
terminali",ylab="cross-validation errors")
plot(cv.data_cancer$k,cv.data_cancer$dev,      type="b",col="#1E77E8",xlab="cost-complexity
parameter",ylab="cross-validation errors")
prune.data_cancer=prune.misclass(tree.data_cancer,best=4)
par(mfrow=c(1,1))
plot(prune.data_cancer,col="grey",lwd=3)
text(prune.data_cancer,pretty=1,col="#1E77E8")
tree.pred1=predict(prune.data_cancer,data_cancer.test,type="class")
#table(tree.pred,diagnosis.test)
matrice1=confusionMatrix(tree.pred1,diagnosis.test)
matrice1
prune.data_cancer=prune.misclass(tree.data_cancer,best=5)
par(mfrow=c(1,1))
plot(prune.data_cancer,col="grey",lwd=1.5)
text(prune.data_cancer,pretty=1,col="#1E77E8",cex=0.8)
tree.pred=predict(prune.data_cancer,data_cancer.test,type="class")
#table(tree.pred,diagnosis.test)
confusionMatrix(tree.pred,diagnosis.test)
#bagging
```

```
library(randomForest)
set.seed(4)
bag.data_cancer=randomForest(diagnosis~.,data_cancer,mtry=30,subset=train,importance=T)
bag.data_cancer
#plot(bag.data_cancer)
oob.error.data=data.frame(Trees=rep(1:nrow(bag.data_cancer$err.rate),times=3),
                          Type=rep(c("OOB","B","M"),
                                   each=nrow(bag.data_cancer$err.rate)),

Error=c(bag.data_cancer$err.rate[,"OOB"],bag.data_cancer$err.rate[,"B"],bag.data_cancer$err.r
ate[,"M"]))
ggplot(data=oob.error.data,aes(x=Trees,y=Error))+geom_line(aes(color=Type))
bag.data_cancer.test=data_cancer[-train,]
bag.diagnosis.test=data_cancer$diagnosis[-train]
tree.pred2=predict(bag.data_cancer,bag.data_cancer.test,type="class")
#table(tree.pred,bag.diagnosis.test)
matrice2=confusionMatrix(tree.pred2,diagnosis.test)
matrice2
library(car)
colors <- c("#56B4E9", "#E69F00")
scatterplotMatrix(data_cancer[,2:5],groups=data_cancer$diagnosis,lower.panel=NULL,col=colo
rs)
cor(data_cancer[-1])
#randomForest
set.seed(1)
rf.data_cancer=randomForest(diagnosis~.,data_cancer,subset=train,mtry=5,importance=T)
rf.data_cancer
oob.error.data=data.frame(Trees=rep(1:nrow(rf.data_cancer$err.rate),times=3),
            Type=rep(c("OOB","B","M"),
            each=nrow(rf.data_cancer$err.rate)),

Error=c(rf.data_cancer$err.rate[,"OOB"],rf.data_cancer$err.rate[,"B"],rf.data_cancer$err.rate[,"
M"]))
ggplot(data=oob.error.data,aes(x=Trees,y=Error))+geom_line(aes(color=Type))
#plot(rf.data_cancer)
rf.data_cancer.test=data_cancer[-train,]
rf.diagnosis.test=data_cancer$diagnosis[-train]
tree.pred3=predict(rf.data_cancer,rf.data_cancer.test,type="class")
#table(tree.pred,rf.diagnosis.test)
matrice3=confusionMatrix(tree.pred3,diagnosis.test)
matrice3
importance(rf.data_cancer)
varImpPlot(rf.data_cancer,col="#1E77E8",main="Misure di importanza delle variabili")
#boosting
library(gbm)
set.seed(1)
data_cancer$diagnosis=as.numeric(data_cancer$diagnosis)-1
boost.data_cancer=gbm(diagnosis~.,data_cancer[train,],shrinkage=0.01,distribution
"bernoulli",n.trees=5000,interaction.depth=4)
summary(boost.data_cancer)
```

```
#plot(boost.data_cancer,i="radius_worst")
boost.data_cancer.test=data_cancer[-train,]
boost.diagnosis.test=data_cancer$diagnosis[-train]
tree.pred4=predict.gbm(boost.data_cancer,boost.data_cancer.test,n.trees = 5000,type = "response")
#table(tree.pred,boost.diagnosis.test)
tree.pred4=as.numeric(tree.pred4 > 0.5)
matrice4=confusionMatrix(factor(tree.pred4),factor(boost.diagnosis.test))
matrice4
decision_tree=matrice1$overall[c(1,3,4)]
baggin
g_tree=matrice2$overall[c(1,3,4)]
randomforest=matrice3$overall[c(1,3,4)]
Boosting_tree=matrice4$overall[c(1,3,4)]
decision_tree
bagging_tree
randomforest
Boosting_tree
boxplot(decision_tree,bagging_tree,randomforest,Boosting_tree,names=c("decision tree","Bagging","randomforest","Boosting"),col="#1E77E8",border = "#E56E45")


df=read.csv2("/Users/leonardomichi/Desktop/winequality-red.csv",header=T,sep=";",dec=".")
good=factor(ifelse(df$quality<=5,"no","yes"))
df=data.frame(df,good)
df=df[-12]
summary(df)
cor(df[-12])
colors <- c("#81F749", "#4983F7")
scatterplotMatrix(df[,1:4],groups = df$good,col = colors,lower.panel=NULL)
#hist(df$citric.acid)
#shapiro.test(df$citric.acid)
set.seed(5)
#regressione logistica
train=sample(1:nrow(df),1000)
df.test=df[-train,]
good.test=df$good[-train]
log.reg=glm(good~volatile.acidity+free.sulfur.dioxide+total.sulfur.dioxide+sulphates+alcohol,data=df,family = binomial,subset = train)
summary(log.reg)
coef(log.reg)
log.pred=predict(log.reg,df.test,type = "response")
log.pred[1:10]
contrasts(good)
pred=rep("no",599)
pred[log.pred>0.5]="yes"
matrice1=confusionMatrix(as.factor(pred),good.test)
#table(pred,good)

#linear disciminant analysis
```

```
library(MASS)
lda.fit=lda(good~volatile.acidity+free.sulfur.dioxide+total.sulfur.dioxide+sulphates+alcohol,dat
a=df,subset = train)
lda.fit
plot(lda.fit)
lda.pred=predict(lda.fit,df.test)
lda.class=lda.pred$class
matrice2=confusionMatrix(lda.class,good.test)
#ldahist(lda.pred$x,g=df$good)
#Naive Bayes
library(e1071)
nb.fit=naiveBayes(good~volatile.acidity+free.sulfur.dioxide+total.sulfur.dioxide+sulphates+alc
ohol,data=df,subset = train)
nb.fit
nb.class=predict(nb.fit,df.test)
matrice3=confusionMatrix(nb.class,good.test)
#alberi
#decision tree
#tree.fit=tree(good~.,df,subset=train)
#summary(tree.fit)
#tree.pred=predict(tree.fit,df.test,type="class")
#confusionMatrix(tree.pred,good.test)
library(randomForest)
#bagging
bag.fit=randomForest(good~.,df,mtry=11,subset=train,importance=T)
bag.fit
bag.pred=predict(bag.fit,df.test,type="class")
matrice4=confusionMatrix(bag.pred,good.test)
#randomforest
rf.fit=randomForest(good~.,df,subset=train,mtry=3,importance=T)
rf.fit
#oob.error.data=data.frame(Trees=rep(1:nrow(rf.fit$err.rate),times=3),
 #                        Type=rep(c("OOB","Yes","No"),
  #                               each=nrow(rf.fit$err.rate)),
   #                      Error=c(rf.fit$err.rate[,"OOB"],rf.fit$err.rate[,"Yes"],rf.fit$err.rate[,"No"]))
#ggplot(data=oob.error.data,aes(x=Trees,y=Error))+geom_line(aes(color=Type))
rf.pred=predict(rf.fit,df.test,type="class")
confusionMatrix(rf.pred,good.test)
importance(rf.fit)
varImpPlot(rf.fit,col="#1E77E8",main="Misure di importanza delle variabili")
#boosting
library(gbm)
df$good=as.numeric(df$good)-1
good.test=df$good[-train]
boost.fit=gbm(good~.,df[train,],shrinkage=0.01,distribution                                    =
"bernoulli",n.trees=5000,interaction.depth=4)
summary(boost.fit)
boost.pred=predict.gbm(boost.fit,df.test,n.trees = 5000,type = "response")
boost.pred=as.numeric(boost.pred>0.5)
confusionMatrix(factor(boost.pred),factor(good.test))
```

```
reg.log=matrice1$overall[c(1,3,4)]
lda=matrice2$overall[c(1,3,4)]
Bayes=matrice3$overall[c(1,3,4)]
random.forest=matrice4$overall[c(1,3,4)]
boxplot(reg.log,lda,Bayes,random.forest,names=c("regressione logistica","LDA","naive
Bayes","random forest"),col="#1E77E8",border = "#E56E45")
```

BIBLIOGRAFIA:

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An introduction to statistical learning (Second Edition). New York: Springer.
2. Friedman, J., Hastie, T., & Tibshirani, R. (2013). The elements of statistical learning. Second edition. Springer, Berlin: Springer series in statistics.
3. Breiman, L. Random Forests. *Machine Learning* **45**, 5–32 (2001).
4. Biau, G., Scornet, E. A random forest guided tour. *TEST* **25**, 197–227 (2016).
5. Wolberg,William, Mangasarian,Olvi, Street,Nick, and Street,W.. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository.
6. Cortez,Paulo, Cerdeira,A., Almeida,F., Matos,T., and Reis,J.. (2009). Wine Quality. UCI Machine Learning Repository.
7. Introduzione ai Modelli Statistici Giovanni M. Marchetti Dipartimento di Statistica, Informatica, Applicazioni, Firenze 2013, 2019 rev. 1
8. Wasserman, L. (2004). All of statistics. Springer-Verlag, New York.

SITOGRAFIA:

- https://doi.org/10.1023/A:1010933404324
- https://hastie.su.domains/ISLR2/ISLRv2_website.pdf
- https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12_toc.pdf
- https://doi.org/10.1007/s11749-016-0481-7
- https://www.usu.edu/math/adele/randomforests/ovronnaz.pdf
- https://doi.org/10.24432/C5DW2B.
- https://doi.org/10.24432/C56S3T.
- https://www.researchgate.net/profile/Giovanni-Marchetti-2/publication/335320657_Introduzione_ai_Modelli_Statistici/links/5d5e5038458515210257d11e/Introduzione-ai-Modelli-Statistici.pdf

Ringraziamenti: