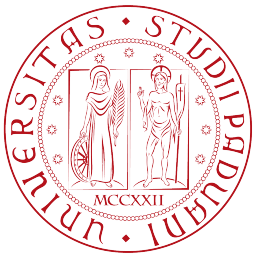


UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Matematica

Tecniche di Machine Learning per la Selezione della Terapia nei casi di Malaria Severa

Relatore
Prof. Francesco Rinaldi

Laureando
Luca Dal Zotto
Numero di matricola
1142645

27 Settembre 2019 - Anno Accademico 2018/2019

Indice

Capitolo 1

Introduzione

1.1 Presentazione del problema

La malaria è tra le più importanti malattie infettive al mondo per diffusione e mortalità. Globalmente, circa 3,3 miliardi di persone vivono in aree endemiche, in cui è presente la zanzara *Anofele*, principale vettore del plasmodio della malaria. Si stima che nel 2016 i casi accertati di malaria siano stati circa 216 milioni, fra cui 445.000 decessi [1]. La grandezza di tali numeri lascia subito intuire l'eterogeneità del fenomeno e quindi il conseguente bisogno di una classificazione particolareggiata dei pazienti.

Una prima complicazione riguarda i casi di **malaria severa**. Tra essi una diagnosi precisa e una scelta corretta del trattamento diventano di cruciale importanza per raggiungere il duplice obiettivo di minimizzare il tasso di mortalità e tenere contenute le spese mediche: si tenga presente che le regioni dove la malaria è maggiormente diffusa sono aree povere dell'Africa centrale e di altri paesi tropicali. Nei paesi Europei, non essendo questi endemici, la preoccupazione principale riguarda i casi di **malaria importata**. Nel 2017, in 30 Paesi dell'Unione Europea e dello Spazio Economico Europeo sono stati confermati 8.393 casi di malaria, pari a 1,2 casi per 100.000 abitanti [2].

L'insieme dei dati su cui si basa questa tesi proviene dall'Istituto Nazionale per le Malattie Infettive Lazzaro Spallanzani-IRCCS-Roma. Tale **dataset** consiste in una lista dei principali parametri clinici di 259 pazienti, prevalentemente residenti in Italia. Dei 119 che hanno contratto la malaria severa, è indicata la tipologia di trattamento affrontata: **orale** o **endovenosa**. Differenziando la terapia, oltre a curare il paziente, si mira, ove possibile, a farlo nel modo meno drastico possibile, cioè attraverso cure meno invasive.

Il proposito di questa tesi è quello di creare un modello matematico che, a partire dalle caratteristiche del soggetto, riesca ad individuare la tipologia di trattamento della malaria severa più adatta. Per conseguire questo risultato, il primo passo è quello di trattare ogni paziente come un vettore di $\mathbb{R}^k$, ove k è il numero di parametri considerati (**feature**) a lui

relativi. In questo caso, le feature scelte sono 38 e comprendono alcuni dati demografici, sanitari e altri relativi ai trattamenti anti-malaria affrontati. L'obiettivo è approssimare nel miglior modo possibile la funzione f (non conosciuta) che mappa ogni paziente nel suo trattamento corretto. Dal momento che la variabile dipendente può assumere due valori (terapia orale o endovenosa), il problema così delineato rientra nel campo della **classificazione binaria**. Gli strumenti che qui si useranno per risolverlo sono alcune delle più famose tecniche di **Machine Learning** e nelle prossime sezioni ne viene presentato brevemente il funzionamento.

1.2 Alcuni campi del Machine Learning

Siano X e Y due insiemi. Poco sopra è stato introdotto il problema proprio del formalismo matematico di approssimare una funzione ideale $f : X \to Y$, che descriva un determinato fenomeno. Nel caso in questione, un elemento x del primo insieme è un vettore che presenta in ogni entrata una determinata informazione relativa ad un paziente. La funzione f assegna ad un tale input l'uscita scalare 0 o 1, rappresentante la terapia, perciò il codominio sarà $Y = \{0, 1\}$. In generale si dispone di un insieme D, il dataset, composto da n coppie input-output $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ tali per cui

$$y_i = f(x_i), \quad \forall i = 1, \ldots, n.$$

Questi dati, derivanti dall'osservazione del fenomeno in studio, sono squisitamente *esogeni* e costituiscono la materia prima per gli algoritmi di Machine Learning.

Una possibile distinzione delle categorie (anche dette paradigmi) del Machine Learning prevede la divisione tra **apprendimento supervisionato** e **apprendimento non supervisionato** a seconda dell'utilizzo o meno degli output nella definizione del modello. Nel caso supervisionato, le tecniche in questione agiscono generalmente in due fasi:

- **Fase di apprendimento**: viene utilizzato un sottoinsieme T del dataset (*training set*) per ricavare informazioni sul fenomeno studiato, cercando di estrarre una regola generale che associ l'input all'output corretto;
- **Fase di test**: vengono usati dei dati non coinvolti nella prima fase (*testing set*) per mettere alla prova l'efficacia del modello creato, confrontando l'output previsto con quello effettivo.

Scopo dell'addestramento non è quello di interpolare i dati di training, quanto piuttosto quello di modellare il processo che ha generato i dati. Ciò implica che la scelta dell'architettura deve tener conto dell'esigenza di assicurare buone capacità di generalizzazione. Dal punto di vista teorico, uno dei problemi più importanti è quello di definire opportunamente la complessità del modello: generalmente, una stima troppo complessa non è in grado di classificare bene punti non compresi nell'insieme di addestramento (tale fenomeno è chiamato **overfitting** o overtraining). D'altra parte, modelli troppo semplici spesso non

sono in grado di classificare correttamente tutti i campioni dell'insieme d'addestramento (in questo caso si parla di **underfitting**).

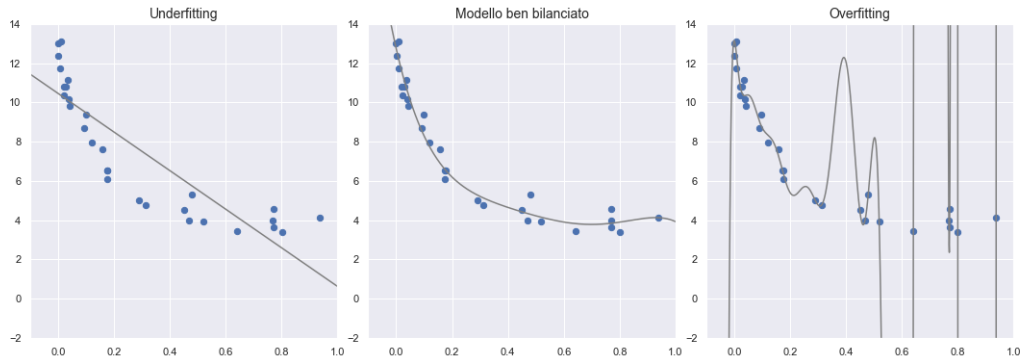


Figura 1.1: Un esempio di underfitting, overfitting e di un modello adeguato.

Nel valutare l'effettiva capacità di generalizzazione del modello può succedere che il sottoinsieme d'addestramento non rappresenti efficacemente l'intero dataset, contenendo elementi con caratteristiche particolari. A superamento di ciò, interviene la **cross validation**: tale procedura consiste nel partizionare il dataset in k parti uguali: di esse $k-1$ vengono coinvolte nella fase di addestramento, mentre la rimanente va a costituire l'insieme di test. L'intero processo viene ripetuto per tutte le k scelte possibili della parte omessa dal training set. Calcolando una media dei risultati ottenuti di volta in volta, si ottiene un punteggio che valuti il modello creato. Il vantaggio della cross validation è quello di aver utilizzato ad ogni step una frazione abbastanza grande del dataset per l'apprendimento, aspetto che dovrebbe garantire una miglior fedeltà al fenomeno.

1.2.1 Dal neurone formale alle reti neurali

Il modello di apprendimento supervisionato più elementare, usato come unità base di calcolo per strumenti più complessi, è il **neurone formale**. Nella sua formulazione più significativa, ovvero il **Perceptron** proposto da Rosenblatt, il vettore in input viene moltiplicato scalarmente con il vettore dei pesi, $w \in \mathbb{R}^k$, e il risultato di tale operazione viene confrontato con un certo valore soglia $\theta \in \mathbb{R}$. A questo punto interviene la funzione di attivazione del neurone g: in base ai casi può essere scelta la funzione segno o l'altrettanto intuitiva funzione a gradino di Heaviside:

$$g(t) := \begin{cases} 1 & \text{se } t \geqslant 0 \\ 0 & \text{se } t < 0 \end{cases}.$$

Questa determina l'uscita del neurone:

$$y(x) \coloneqq g\left(\sum_{j=1}^{k} w_j x^j - \theta\right) \equiv g(w^T x - \theta).$$

L'apprendimento consiste nel determinare i valori dei pesi e della soglia tali da far combaciare, se possibile, il risultato del neurone con l'output effettivo, più precisamente deve verificarsi:

$$\begin{array}{llll} w^\top x_i - \theta \geqslant 0 & \text{se} & y_i = 1, & \\ w^\top x_i - \theta < 0 & \text{se} & y_i = 0. & (i = 1, \ldots, n) \end{array} \tag{1.1}$$

Posto $A = \{x_i : (x_i, y_i) \in T, y_i = 1\}, B = \{x_i : (x_i, y_i) \in T, y_i = 0\}$ la (1.1) può essere interpretata geometricamente come la ricerca di un iperpiano H che separi questi 2 insiemi. Con questo approccio è facile dedurre che il problema in questione ammette soluzione se e solo se A e B sono linearmente separabili. Questa prima evidente limitazione del perceptron suggerisce di studiare modelli più complessi che, eventualmente, coinvolgano i neuroni formali, creando tra di essi dei collegamenti che permettano applicazioni a problemi più generali. Questa idea è alla base della teoria delle **reti neurali**: architetture costituite da più strati di neuroni connessi tra loro.

Una rete neurale multistrato è formata da un primo livello di nodi di ingresso, sprovvisti di capacità di elaborazione, associati agli ingressi della rete, ed $L \geqslant 2$ strati. Di essi $L - 1$ sono costituiti da neuroni che non comunicano verso l'esterno (per questo tali strati sono chiamati **hidden layers**) e il rimanente strato è formato da neuroni le cui uscite rappresentano quelle della rete stessa.

Le connessioni interneurali e le connessioni con i nodi di ingresso sono rappresentate da archi orientati e pesati. In generale si suppone che non vi sia alcuna connessione fra i neuroni di uno stesso livello. Nel caso in cui ogni neurone in uno strato sia connesso a tutti e soli quelli del livello successivo, garantendo così la trasmissione di informazioni in un'unica direzione, si parla di reti **feed forward**. Una struttura di questo genere è illustrata nella Figura 1.2.

Le reti neurali consentono, sotto ipotesi opportune sulle funzioni di attivazione dei neuroni, di approssimare con la precisione voluta una qualsiasi funzione continua su un insieme compatto e quindi, in particolare, di risolvere problemi di classificazione di insiemi non separabili linearmente [3]. La determinazione dei parametri attraverso il processo di addestramento diviene tuttavia un problema più complesso e richiede l'impiego di tecniche di ottimizzazione non lineare per risolvere un problema del tipo:

$$\min_{w \in \mathbb{R}^k} E(w) = \min_{w \in \mathbb{R}^k} \sum_{i=1}^{n} E_i(w), \tag{1.2}$$

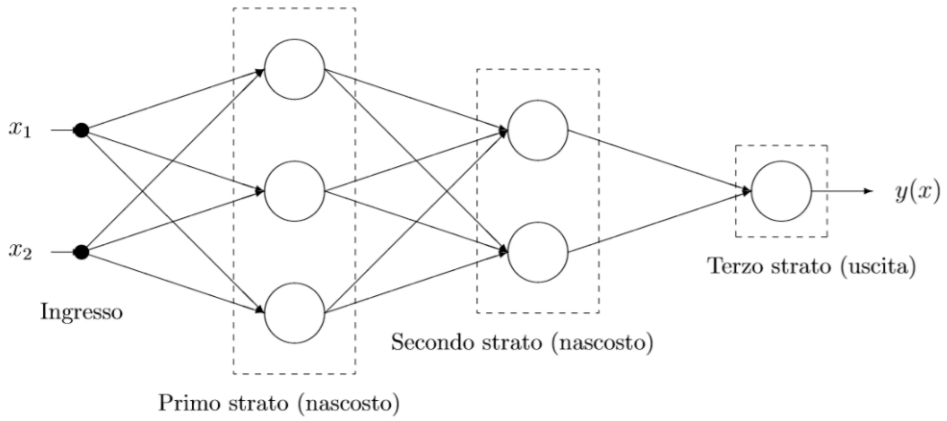


Figura 1.2: Esempio di rete neurale a 3 strati, con 2 strati nascosti, 2 ingressi, 1 uscita.

in cui E_i è il termine di errore relativo al i-esimo campione e misura la distanza tra l'uscita desiderata y_i e l'uscita $y(x_i; w)$ fornita dalla rete. La misura più usata è l'errore quadratico:

$$E_i(w) = \frac{1}{2}\|y(x_i; w) - y_i\|^2,$$

ma è possibile considerare anche funzioni di errore di struttura diversa. Uno dei primi algoritmi proposti per il calcolo dei pesi in una rete neurale è il metodo di **backpropagation**, il quale fa uso della discesa del gradiente. L'algoritmo di apprendimento può essere diviso in due fasi:

1. **propagazione:** un vettore input viene fornito alla rete e propagato in avanti, strato per strato, fino a raggiungere il livello di output. Successivamente, viene calcolato l'errore;
2. **aggiornamento del peso:** applicando il metodo di discesa del gradiente sulla funzione errore, vengono gradualmente modificati i pesi di ogni neurone a partire da quelli nello strato di output e arrivando, procedendo a ritroso, a quelli del primo strato [4] [5].

1.2.2 Selezione delle Feature

Il punto di partenza per l'apprendimento dei vari algoritmi di Machine Learning sono le informazioni racchiuse nel dataset. Per comprendere come contribuiscono al processo le varie feature, ovvero le componenti dei vettori input, bisogna innanzitutto considerare il fatto che solo alcune di esse hanno un ruolo determinante, mentre altre possono rivelarsi un fattore di disturbo, arrivando a costituire persino un ostacolo per il classificatore.

In un primo approccio, è possibile dividere le feature in:

- **Irrilevanti:** non aggiungono informazioni utili all'apprendimento dando origine, in alcuni casi, a del rumore;
- **Ridondanti:** il loro contributo non è rilevante ai fini della classificazione, perché l'informazione racchiusa è già stata fornita da altre feature;
- **Rilevanti:** né irrilevanti né ridondanti.

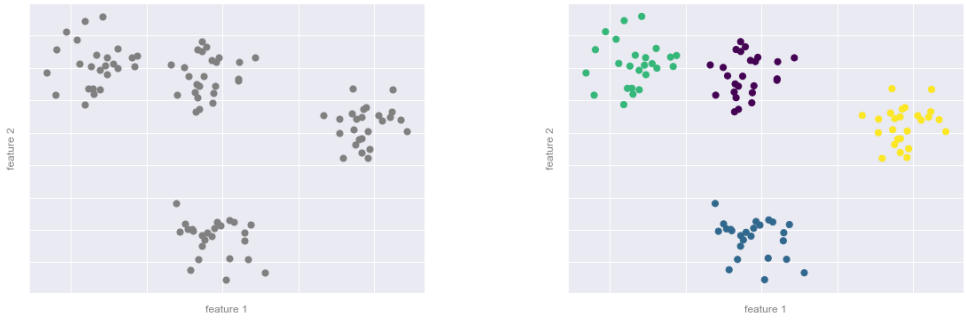
Escludendo rare eccezioni, a priori non è possibile identificare le feature rilevanti tra quelle a disposizione. Per evitare la perdita di informazioni, si può allora pensare di considerare la totalità delle feature disponibili. Tuttavia, oltre al già anticipato problema di presenza di eventuale rumore con relativa perdita di capacità di generalizzazione del classificatore, si rischia di aumentare il costo computazionale, in quanto il numero di feature coincide con la dimensionalità del problema. Per ovviare a queste conseguenze indesiderate, si ricorre a strumenti di Machine Learning in grado di individuare un sottoinsieme minimale di feature che garantisca buone capacità di generalizzazione, ovvero effettuare una **Feature Selection**. Le tecniche esistenti vengono divise in 3 categorie:

- **Metodi Filtro:** usano una funzione che assegni un punteggio ad ogni feature valutandone la capacità predittiva, permettendo così di stilare una classifica in base all'importanza; le feature con il punteggio più alto saranno tenute in considerazione, mentre quelle peggiori verranno eliminate dal dataset. Un aspetto importante è che tale pre-processamento dei dati è indipendente dall'algoritmo di predizione;
- **Metodi Wrapper:** consistono nel considerare varie combinazioni delle feature, valutarle e compararle tra di loro per sceglierne il sottoinsieme migliore. A differenza dei metodi filtro, qui viene usato un determinato modello di apprendimento nella fase di valutazione;
- **Metodi Embedded:** individuano le feature che contribuiscono maggiormente all'accuratezza del modello durante il processo di addestramento e perciò sono dipendenti dall'algoritmo scelto [6].

1.2.3 Apprendimento non supervisionato

Con l'espressione apprendimento non supervisionato, si intende una categoria di tecniche di apprendimento che, a partire unicamente dalle feature, deducono informazioni sui dati sulla base di caratteristiche comuni, senza sapere effettivamente a che classe appartiene un certo elemento. In un certo senso, lo scopo è fare in modo che il dataset "parli da sé". Una classe di tecniche d'apprendimento non supervisionato è il **Clustering** o analisi dei gruppi. L'obiettivo è trovare analogie tra i dati che permettano una partizione del dataset: ciascun gruppo (cluster) che si viene a formare deve contenere degli elementi simili (in un senso da definire) tra loro, ma differenti da quelli degli altri gruppi. L'efficacia si riconosce nel

momento in cui i vari cluster sono ben separati tra loro. Si consideri ad esempio l'insieme dei punti rappresentati in Figura 1.3.a:



(a) *Dati originali* (b) *Dati suddivisi in cluster*

Figura 1.3: Esempio del funzionamento di un algoritmo di clustering.

A prima vista, in base ai valori assunti dalle due feature considerate (da interpretare come due coordinate spaziali), si intuisce l'esistenza di una suddivisione in 4 parti del dataset, ad esso intrinseca. Da un algoritmo di clustering ci si aspetta che assegni ad ogni campione un'etichetta che identifichi il cluster di appartenenza. Nella Figura 1.3.b sono stati evidenziati i quattro cluster individuati attraverso quattro colori distinti.

In base alla geometria del dataset e al numero di cluster in questione, il problema può assumere diverse sfaccettature. Per prima cosa è necessario formalizzare matematicamente il concetto di somiglianza tra due elementi. A tal scopo si introduce una metrica: generalmente allo spazio $\mathbb{R}^k$ si associa la classica metrica euclidea, perció due elementi x e y saranno tanto più simili quanto più piccola è la loro distanza:

$$d(x, y) = \|x - y\|_2 = \sqrt{\sum_{j=1}^{k} (x^j - y^j)^2}.$$

La scelta della metrica può cambiare in base al tipo di dati in analisi. Le tecniche di clustering si basano principalmente su due filosofie:

- **Metodi Aggregativi** (dal basso verso l'alto)**:** inizialmente tutti gli elementi sono considerati cluster a sé stanti, e successivamente l'algoritmo provvede ad unire quelli più vicini fino ad ottenere un numero prefissato di cluster, oppure fino a che la distanza minima tra di essi non supera un certo valore;
- **Metodi Divisivi** (dall'alto verso il basso)**:** all'inizio l'intero dataset è considerato come un unico cluster che viene progressivamente suddiviso in modo da ottenere gruppi sempre più omogenei.

Un'altra importante classificazione è quella che tiene conto del metodo usato per dividere lo spazio. Si distingue tra:

- **Clustering Partizionale:** considerando la distanza da dei punti rappresentativi dei cluster (*centroidi*), viene stabilita l'appartenenza di ciascun elemento ad esattamente un cluster;
- **Clustering Gerarchico:** viene permessa la definizione di sotto-cluster, costruendo una gerarchia di sottoinsiemi, visualizzabile mediante una rappresentazione ad albero (*dendrogramma*), in cui sono rappresentati i passi di accorpamento/divisione dei gruppi.

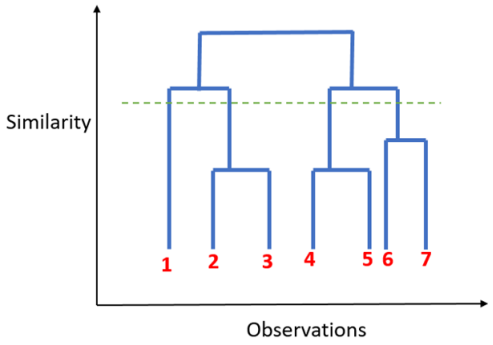


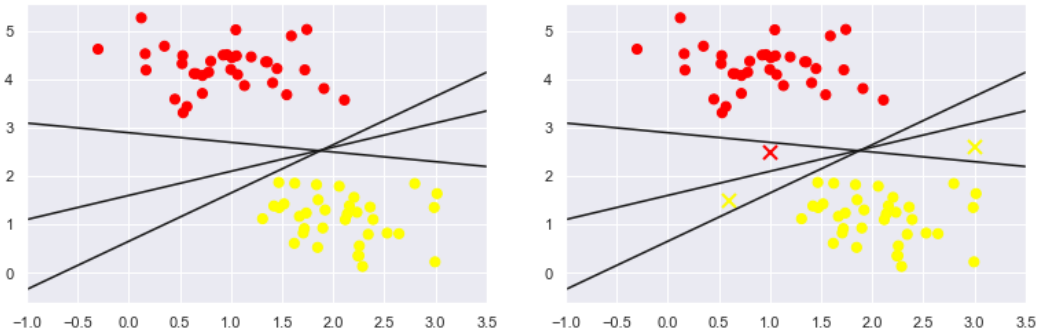
Figura 1.4: Dendogramma di un algoritmo di Clustering Gerarchico.

Capitolo 2

Strumenti esistenti nel Machine Learning

2.1 Support Vector Machine

Nel capitolo precedente è stato introdotto il funzionamento del perceptron e il suo obiettivo: trovare un iperpiano che separi linearmente il dataset per poi usarlo come classificatore binario. Un aspetto importante da comprendere è che esiste un numero infinito di vettori peso w che soddisfano la (1.1) (rispettivamente, esistono infiniti iperpiani separatori). Inoltre, andando ad analizzare nel dettaglio l'algoritmo del percepron, si può notare come il vettore dei pesi venga inizializzato casualmente e, di conseguenza, eseguendo più volte l'algoritmo si possono ottenere output differenti. Nella Figura 2.1.a sono raffigurati 3 possibili iperpiani individuati dal perceptron su un dataset di esempio.



(a) *Esempio di 3 iperpiani separatori*

(b) *Le croci rappresentano dei dati non appartenenti al training set*

Figura 2.1: Possibili risultati del Perceptron.

A primo impatto, l'esistenza di più soluzioni potrebbe non sembrare un problema, in quanto tutti e tre gli iperpiani separano perfettamente i dati. Tuttavia quando si usa un algoritmo di apprendimento come questo, lo scopo finale non è classificare perfettamente i dati di cui si è già in possesso, bensì è classificare correttamente nuovi dati, non appartenenti al training set. Invece, per quanto detto, il perceptron non dà garanzie sulla capacità di generalizzazione del modello. Infatti, continuando l'esempio grafico precedente, si può notare come, introducendo dei nuovi dati, due dei tre iperpiani trovati non riescano a classificarli correttamente (Figura 2.1.b).

Ci si domanda quindi se sia possibile, in fase di addestramento, capire quale sia l'iperpiano migliore. I modelli di apprendimento di **Support Vector Machine** (SVM) offrono una possibile strada per rispondere a questo interrogativo. Per prima cosa serve un indice che permetta di comparare tra loro gli iperpiani. L'idea è quella di dare alle linee un certo spessore, disegnando cioè delle regioni lungo gli iperpiani fino al punto (dell'insieme di addestramento) più vicino, detto **vettore di supporto**. Ecco un esempio:

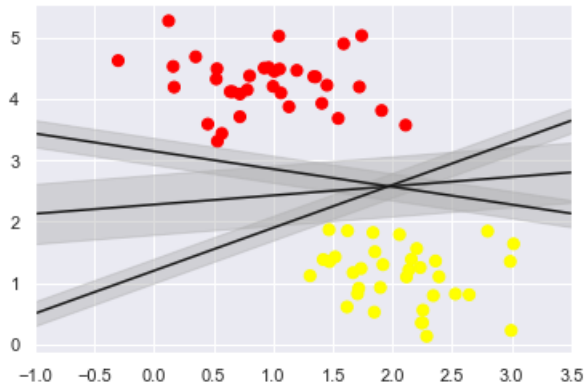


Figura 2.2: Aggiunta del margine di tolleranza ai 3 iperpiani.

Come evidenziato da alcuni risultati della teoria statistica dell'apprendimento [7], maggiore è la distanza dell'iperpiano dal punto più vicino di ognuna delle classi, minore è l'errore di generalizzazione del classificatore. Per questo, in SVM l'iperpiano che massimizza questo margine di errore è quello ottimo per il modello in studio. Inoltre, è possibile dimostrarne l'esistenza e l'unicità [8].

2.1.1 Analisi matematica dell'algoritmo di apprendimento

Si considerino due insiemi disgiunti di punti A e B in $\mathbb{R}^k$, e si assuma che siano linearmente separabili. Sia P la cardinalità di $A \cup B$. Dato un iperpiano di separazione $H = \{x \in \mathbb{R}^k : w^T x + \theta = 0\}$, si introduce il

Definizione 2.1.1. Il **margine di separazione** ρ dell'iperpiano H è la minima distanza tra i punti in $A \cup B$ e l'iperpiano H, cioè:

$$\rho(w,\theta) = \min_{x_i \in A \cup B} \left\{ \frac{\left| w^T x_i + \theta \right|}{\|w\|} \right\}.$$

Per quanto detto, determinare l'iperpiano ottimo equivale a risolvere il problema:

$$\begin{array}{ll} \max_{w,\theta} & \rho(w,\theta) \\ s.t. & w^T x_i + \theta \geq 1, \quad x_i \in A, \\ & w^T x_i + \theta \leq -1, \quad x_i \in B. \end{array} \tag{2.1}$$

A sua volta, si può dimostrare [8] che il problema (2.1) è equivalente al problema di programmazione quadratica convessa:

$$\begin{array}{ll} \min_{w,\theta} & \frac{1}{2}\|w\|^2 \\ s.t. & y_i \left(w^T x_i + \theta \right) - 1 \geq 0, \quad (i = 1, \ldots, P) \end{array} \tag{2.2}$$

in cui $y_i = +1$ se $x_i \in A$ e $y_i = -1$ se $x_i \in B$.

La forma standard di questa classe di problemi di ottimizzazione è la seguente:

$$\begin{array}{ll} \min & \frac{1}{2} x^T Q x + c^T x \\ s.t. & Ax \leq b, \end{array}$$

con $x \in \mathbb{R}^k, Q \in \mathbb{R}^{k \times k}, c \in \mathbb{R}^k, A \in \mathbb{R}^{m \times k}$ e $b \in \mathbb{R}^m$.

Per la risoluzione, spesso si ricorre alla teoria della dualità, dalla quale segue che il valore ottimo del problema (2.2), detto problema primale, è uguale al valore ottimo del problema duale. Definita la funzione Lagrangiana

$$L(w,\theta,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{P} \alpha_i \left[y_i \left(w^T x_i + \theta \right) - 1 \right], \quad \alpha_i \geq 0, \quad i = 1, \ldots, P$$

al (2.2) si può associare il problema duale di Wolfe [9]:

$$\begin{aligned} \max_{w,\theta,\alpha} L(w,\theta,\alpha) = & \frac{1}{2}\|w\|^2 - \sum_{i=1}^{P} \alpha_i \left[y_i \left(w^T x_i + \theta \right) - 1 \right] \\ & w = \sum_{i=1}^{P} \alpha_i y_i x_i \\ & \sum_{i=1}^{P} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, \ldots, P. \end{aligned} \tag{2.3}$$

Infine, questo può essere riscritto come:

$$\begin{aligned}\min_{\alpha} W(\alpha) =& \frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P} y_i y_j \left(x_i\right)^T x_j \alpha_i \alpha_j - \sum_{i=1}^{P}\alpha_i \\ & \sum_{i=1}^{P}\alpha_i y_i = 0 \quad \alpha_i \geq 0 \quad i = 1, \ldots, P.\end{aligned} \tag{2.4}$$

Il motivo per cui si è arrivati a questa forma è che nella risoluzione del (2.1) è necessario calcolare il prodotto scalare $w^T x$ per ogni nuovo punto da classificare, processo spesso dispendioso dal punto di vista computazionale.

A questo punto, si calcola la soluzione ottima $\bar{\alpha}$ di (2.4). Il vettore $\bar{w}$ che risolve (2.3) sarà:

$$\bar{w} = \sum_{i=1}^{P} \bar{\alpha_i} y_i x_i.$$

Si può verificare che $\bar{\alpha_i}$ è diverso da 0 in corrispondenza dei soli vettori di supporto, dunque $\bar{w}$ dipende solo da questi campioni. A partire da $\bar{w}$ si risale alla soluzione del problema primale (2.2) [10].

2.1.2 Caso non separabile linearmente: metodi kernel

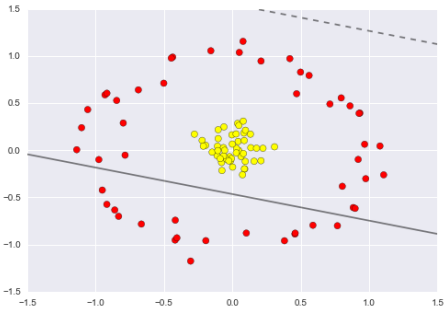
L'efficacia di SVM diventa ancora più evidente quando i dati in studio non permettono una separazione lineare. In questo caso si ricorre ai **metodi kernel**, che consistono nel proiettare i dati in uno spazio di dimensione maggiore (denominato **feature space**) in cui il dataset risulti linearmente separabile. Detta ϕ tale mappa, si introduce la

Definizione 2.1.2. La **funzione kernel** K è definita dal seguente prodotto scalare:

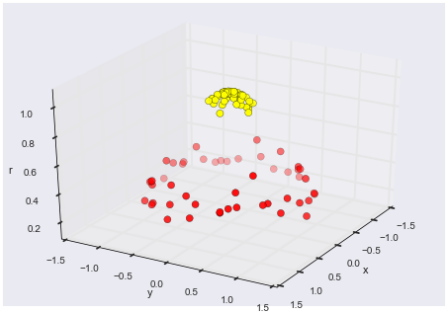
$$K\left(x_i, x_j\right) := \phi\left(x_i\right)^t \phi\left(x_j\right).$$

In relazione ai casi in esame, può essere scelta una particolare funzione kernel. Gli esempi più utilizzati sono:

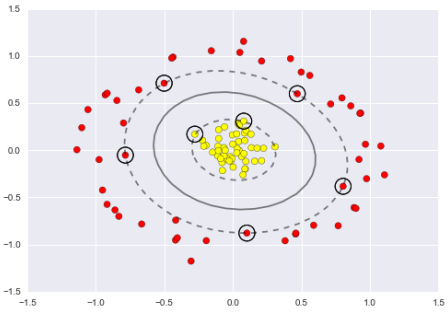
- **Kernel Lineare:** $K\left(x_i, x_j\right) = x_i^t x_j$;
- **Kernel Polinomiale:** $K\left(x_i, x_j\right) = \left(\gamma x_i^t x_j + r\right)^d, \gamma > 0,\ r \geq 0,\ d \in \mathbb{N},\ d \geq 1$;
- **Kernel Gaussiano:** $K\left(x_i, x_j\right) = \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right), \gamma > 0.$



(a) *Kernel Lineare*



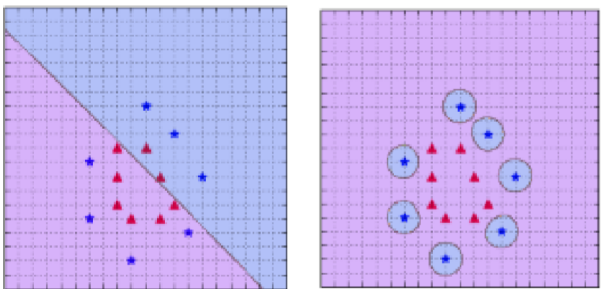
(b) *Proiezione del dataset*



(c) *Kernel RBF e vettori di supporto*

Figura 2.3: Implementazione del Kernel Lineare e del Kernel Gaussiano (detto anche **Funzione di Base Radiale (RBF)**) con un dataset non separabile linearmente.

La costante positiva γ rientra tra i cosiddetti **iperparametri**: in generale, una volta scelta la classe di algoritmi desiderata, è possibile intervenire per rendere il modello adatto al problema particolare, personalizzando i valori di alcune opzioni, chiamate appunto iperparametri. In questo caso, γ controlla la dimensione del kernel: per valori piccoli, il modello si comporta come SVM lineare, mentre per valori troppo grandi, il modello risulterà influenzato fortemente dai vettori di supporto, come mostrato in Figura 2.4.



(a) $\gamma = 10^{-5}$ (b) $\gamma = 2$

Figura 2.4: Conseguenze di valori di γ troppo piccoli o troppo elevati.

2.1.3 Soft-margin SVM

Finora è stato discusso il caso ideale in cui il dataset ammetteva una separazione perfetta (sia essa lineare o meno). Quando lo studio coinvolge problemi reali, tale ipotesi spesso non è più valida. Un modo per trattare queste situazioni è permettere ad alcuni dati di violare il margine di una certa quantità, arrivando se necessario a tollerare errori di classificazione sugli elementi del training set. In questa formulazione, detta **soft-margin SVM**, vengono introdotte delle variabili positive ξ_i, con $i = 1, \dots, n$ e si indebolisce la richiesta di separazione lineare del dataset nel seguente modo:

$$y_i \left(w^\top x_i + \theta \right) \geq 1 - \xi_i. \quad (i = 1, \dots, n)$$

Si noti che per i campioni che si trovano nel bordo del margine, vale $y_i \left(w^\top x_i + \theta \right) = 1$, dunque la quantità ξ_i misura di quanto il campione (x_i, y_i) possa addentrarsi all'interno del margine. Appare quindi naturale concepire $\sum_{i=1}^{n} \xi_i$ come una quantità da minimizzare. Facendo specificare all'utente un parametro $C > 0$ che pesi l'errore di training, si può modificare il problema di minimo (2.2) nel modo seguente:

$$\begin{array}{ll} \min_{w,\theta,\xi} & \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} & y_i \left(w^\top x_i + \theta \right) \geq 1 - \xi_i \quad (i = 1, \dots, n) \\ & \xi_i \geq 0. \end{array} \tag{2.5}$$

Ricordando che il primo addendo della funzione obiettivo riguarda la grandezza del margine mentre il secondo eventuali violazioni, si può intuire la strategia per la scelta dell'iperparametro C: con un C grande si aumenta l'accuratezza sul training set e si accetta un margine più stretto, rischiando, per valori troppo elevati, di dar origine a fenomeni di overfitting, mentre per valori di C piccoli, si massimizza la dimensione del margine, tollerando però violazioni di esso [11].

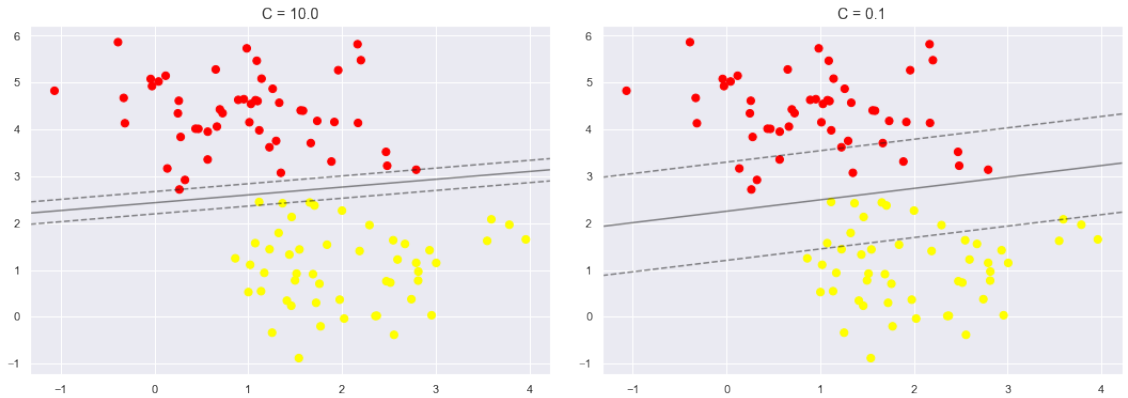


Figura 2.5: Effetto del valore di C sul risultato finale.

2.2 Decision Tree e Random Forest

Un albero di decisione (**decision tree**) è un algoritmo di apprendimento supervisionato che classifica in modo estremamente intuitivo i dati in analisi. L'idea consiste nel partizionare in modo ricorsivo il dataset sulla base di determinate caratteristiche, stabilendo così la struttura del modello. Per esempio, si supponga di avere a disposizione un cesto con della frutta e di chiedere a qualcuno che non l'abbia mai vista prima di classificarla. Questa persona cercherebbe allora di individuare una alla volta delle caratteristiche che differenzino un frutto dall'altro, per esempio distinguendoli inizialmente in base al colore, successivamente per la forma e infine per la dimensione. Questa è effettivamente la filosofia alla base di un decision tree. Un modello così costruito ammette una rappresentazione grafica formata da un insieme di nodi connessi tra loro attraverso dei lati che danno origine ad una struttura ad albero (da qui il nome della classe di algoritmi), dove ogni nodo può presentare un lato entrante e dei lati uscenti. Ci sono tre tipologie di nodi:

- **Nodo Radice:** l'unico dotato unicamente di lati uscenti;
- **Nodi Foglia:** non hanno lati uscenti e terminano la classificazione (chiamati per questo anche nodi terminali o di decisione) assegnando ad un certo input una classe che rappresenta l'output più appropriato;
- **Nodi Interni:** hanno esattamente un lato entrante e almeno due lati uscenti.

Escludendo i nodi foglia, gli altri suddividono il dataset in due o più sottoinsiemi, in base ad una certa funzione dei dati in ingresso. Tale suddivisione può avvenire in relazione al valore assunto da una particolare feature: nel caso di una feature numerica, questo valore per esempio può essere confrontato con una soglia fissata. Per costruire un decision tree è quindi sufficiente scegliere le funzioni di attivazione dei vari nodi.

Per una possibile rappresentazione dell'esempio introdotto precedentemente si veda la Figura 2.6.

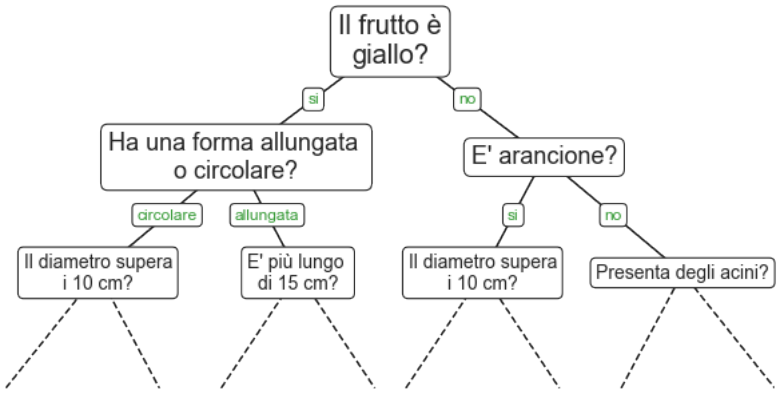


Figura 2.6: Esempio di un albero di decisione per la classificazione della frutta.

Un albero di decisione sarà da preferirsi ad un altro nel momento in cui l'omogeneità delle classi nei sottoinsiemi creati sarà maggiore. Per misurare questa quantità vengono introdotti degli indici di impurità, come l'**indice di Gini**. Scelto S un sottoinsieme formato dopo una certa suddivisione (*split*), l'indice di Gini è definito come:

$$\text{Gini}(S) = 1 - \sum_j p_j^2,$$

dove p_j è la probabilità di estrarre da S un campione appartenente alla j-esima classe (effettiva). Se S è composto prevalentemente da elementi appartenenti alla k-esima classe, si avrà $p_j \approx 0$ se $j \neq k$, mentre $p_j \approx 1$ se $j = k$; di conseguenza l'indice di Gini sarà un valore vicino a zero. Se invece l'insieme S ha una natura più eterogenea, l'indice di Gini sarà maggiore.

Un'alternativa è l'**indice di Entropia**, definito come:

$$\text{Entropia}(S) = \sum_j -p_j \cdot \log_2(p_j).$$

Poichè p_j assume valori tra 0 e 1, $\log_2(p_j)$ sarà negativo. Essendo questo risultato moltiplicato per $-p_j$, si deduce che tutti gli addendi della sommatoria sono positivi, o eventualmente nulli nel caso di una suddivisione perfetta (per $j \neq k$ si ha $p_j = 0$ mentre per $j = k$ si ha $\log_2(p_j) = 0$ dunque in entrambi i casi $-p_j \cdot \log_2(p_j) = 0$). Quindi, come nel caso precedente, minore è il valore dell'indice, migliori sono le suddivisioni trovate [12].

In generale, i vantaggi offerti da un albero di decisione sono:

- facile interpretazione e visualizzazione del risultato;
- eventuali errori possono essere facilmente rintracciati, essendo i vari passaggi semplici e ripercorribili;
- si adatta sia a feature numeriche che categoriche, senza bisogno di normalizzazione;
- ha un buon comportamento su grandi dataset;
- è spesso veloce [13].

Tornando indietro all'esempio sulla frutta, una differenza tra il comportamento di una persona e di un algoritmo sta sul livello di dettaglio necessario per dichiarare terminata la classificazione: una volta raggruppate tutte le mele, banane, ecc., la persona può ritenere terminato il suo compito, mentre una macchina potrebbe continuare a cercare differenze, anche minime all'interno della stessa classe. In ciò consiste il rischio dell'overfitting: aumentando il numero di suddivisioni, si andranno a modellare correttamente i dettagli del training set, perdendo però la capacità di generalizzazione. Un modo per limitare tale fenomeno può essere quello di fissare una profondità massima dell'albero oppure di consentire lo split di un insieme solo se questo presenta un numero sufficiente di elementi.

Le **random forest** (foreste casuali) sono state introdotte per superare i limiti dei decision tree, mantenendone i numerosi pregi. Il loro funzionamento consiste nel combinare i risultati di più alberi. L'idea di unire stimatori che tendono ad avere problemi di overfitting, con lo scopo di ridurre questo effetto indesiderato, è il concetto di fondo ad una tecnica di ricampionamento chiamata *bagging* (bootstrap aggregating). L'algoritmo di random forest consiste nell'addestrare un numero elevato (sull'ordine delle centinaia o migliaia) di decision tree, ognuno dei quali costruito selezionando casualmente alcuni campioni e alcune (in genere $\sqrt{k}$) feature. In questo modo gli alberi saranno scorrelati tra loro, e ciò porterà ad una diminuzione della varianza del modello e del rischio di overfitting [12].

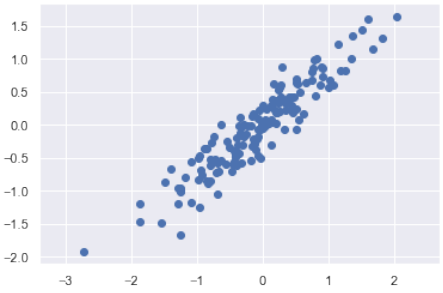
Oltre a fini di classificazione, un modello random forest può essere impiegato utilmente anche per problemi di regressione e di feature selection, rientrando tra i metodi embedded. Per la precisione, in funzione del contributo delle varie feature nel raggiungere il risultato finale, l'algoritmo assegna loro un punteggio, sulla base del quale vengono poi ordinarte in una classifica (**feature ranking**).

2.3 Principal Component Analysis

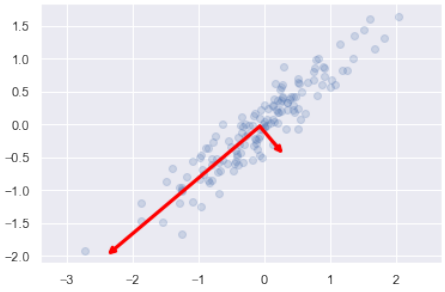
La precedente sezione si è conclusa parlando di come l'algoritmo di random forest possa essere usato allo scopo di individuare le feature più rilevanti per un determinato problema. Il campo del machine learning offre molti altri strumenti in grado di assolvere questo compito. Un esempio, classificato tra i metodi di tipo filtro, è la **selezione delle feature univariata** (UFS). Questo strumento si basa su test che rientrano nel campo della statistica inferenziale per valutare la relazione tra ogni feature e l'output. Una tecnica usata per attribuire un certo valore alle feature è l'analisi della varianza (ANOVA). Essa permette di confrontare due insiemi di dati confrontando la variabilità interna ad essi con la variabilità tra i gruppi. In questo contesto viene introdotto l'ANOVA F-value.

L'**analisi delle componenti principali** (PCA) è un veloce e flessibile metodo non supervisionato impiegato per compiti di riduzione dimensionale. Si consideri a titolo esemplificativo il dataset 2-dimensionale rappresentato in Figura 2.7.a. È chiaro che tra la variabile in ascissa e quella in ordinata ci sia approssimativamente una dipendenza lineare. Analizzando l'intero dataset, PCA individua una lista degli *assi principali*, ovvero dei vettori il cui modulo è direttamente proporzionale alla varianza della distribuzione dei dati lungo quella direzione (Figura 2.7.b).

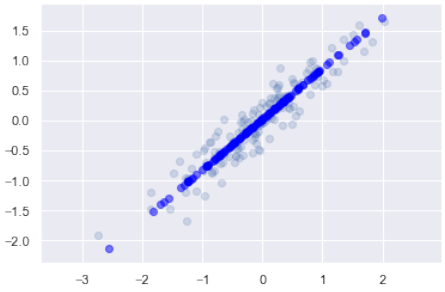
A questo punto, per ottenere una riduzione dimensionale, è sufficiente trascurare le componenti principali meno rilevanti. Nell'esempio introdotto, è possibile osservare come, riducendosi ad una singola dimensione, la relazione tra i punti sia per lo più mantenuta.



(a) *Dati in input*



(b) *Rappresentazione degli assi principali*



(c) *Proiezione dei dati sull'asse principale*

Figura 2.7: Funzionamento di PCA per la riduzione dimensionale.

2.4 K-Means e Spectral Clustering

Un'altra classe di apprendimento non supervisionato è quella degli algoritmi di clustering, introdotta nella sottosezione 1.2.3. Uno dei più conosciuti ed usati è **k-means** e il suo funzionamento può essere così schematizzato:

1. una volta che l'utente ha specificato il numero k di cluster da formare, vengono scelti casualmente k campioni (i centroidi iniziali) tra quelli del dataset;

2. si ripetono i seguenti passaggi finché nessun punto passa da un cluster all'altro o finché i centroidi rimangono gli stessi:

   a) ogni punto viene assegnato al suo centroide più vicino, andando a costituire un cluster;

   b) vengono ricalcolati i centroidi, in base agli elementi appartenenti ai loro cluster. In particolare, il nuovo centroide relativo all'i-esimo cluster sarà la media dei punti di quel cluster.

Una prima osservazione riguarda la fase di inizializzazione: i primi centroidi vengono scelti casualmente, e i risultati finali possono dipendere fortemente da essi. In particolare, non c'è alcuna garanzia riguardo la convergenza dell'algoritmo per tutti i dati iniziali. Per questo

è buona prassi ripetere più volte il processo con scelte iniziali differenti, e individuare il clustering più frequente.

In secondo luogo, per come è definita la seconda fase, segue che l'algoritmo può risultare inefficiente nel caso in cui il dataset abbia geometrie particolari. Si consideri ad esempio quello raffigurato in Figura 2.8. In questo caso bisognerebbe dare a k-means la possibilità di cercare separazioni non lineari, un po' come avviene nei metodi kernel di SVM. A tal proposito è possibile implementare lo stimatore **spectral clustering**.

Spectral clustering trae le sue origini nella teoria dei grafi, dove è utilizzato per cercare sottoinsiemi particolari di nodi (le comunità), usando informazioni relative allo spettro di speciali matrici costruite a partire da un grafo:

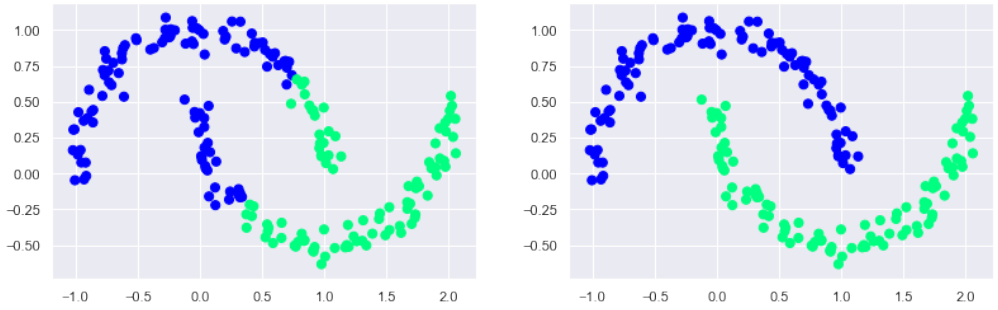
Definizione 2.4.1. La **matrice delle adiacenze** è la matrice la cui entrata (i, j) vale 1 se il nodo i è collegato con un arco al nodo j, 0 altrimenti.

Definizione 2.4.2. La **matrice di grado** è la matrice diagonale avente nell'entrata (i, i) il grado dell'i-esimo vertice del grafo, ovvero il numero di archi collegati ad esso.

Definizione 2.4.3. La **matrice Laplaciana** è la matrice ottenuta facendo la differenza tra la matrice di grado e la matrice delle adiacenze.

Quest'ultima è simmetrica, semidefinita positiva ed ha la proprietà che la dimensione del nucleo corrisponde al numero di componenti connesse del grafo. Il suo primo autovalore non nullo è detto *gap spettrale*, mentre il secondo è detto *valore di Fiedler* ed indica la dimensione del taglio minimo. È possibile dimostrare che l'autovettore relativo all'autovalore di Fiedler racchiude informazioni sulle componenti connesse alle quali appartengono i vari nodi dopo aver rimosso gli archi del taglio minimo per il grafo originale [14].

Il modo più semplice per associare un grafo ad un dataset è costruirlo considerando come nodi i punti del dataset, e tracciando da ognuno di essi m archi verso gli m punti più vicini. Connettendo per esempio ogni punto del dataset in Figura 2.8 ai cinque punti più vicini, è facile intuire che il grafo creato avrà come componenti connesse i due insiemi di punti distribuiti a semiluna.



(a) *Clustering individuato da k-means* (b) *Risultato fornito da spectral clustering*

Figura 2.8: Esempio di dataset la cua geometria impedisce a k-means di individuare i cluster.

Capitolo 3

Tecniche di Machine Learning per la diagnosi

3.1 Preparazione dei dati

Come specificato nell'introduzione, i dati su cui si basa l'analisi di questa tesi sono stati forniti dall'Istituto Nazionale per le Malattie Infettive Lazzaro Spallanzani-IRCCS-Roma. Lo studio si concentra sui 119 pazienti cui è stata diagnosticata la malaria severa, dei quali è nota la terapia somministrata (orale o endovenosa). Il modello matematico che si vuole creare deve riuscire ad individuare la tipologia di trattamento della malaria severa più adatta per ogni individuo sulla base delle feature elencate nella Tabella 3.1.

Criteri ufficiali WHO	**Valori ematici all'atto del ricovero**	**Altro**
cerebral malaria/coma	PLT	età
convulsions	Hb	sesso
acute renal failure	creat	comorbidità
respiratory failure	bil	provenienza
hypoglycaemia	AST	zona in cui si è
shock	ALT	contratta l'infezione
spontaneous bleeding	Na	pregressa malaria
acidosis	parassitemia baseline	durata permanenza
jaundice		nel paese endemico
liver function test		ritardo diagnosi
>3 time normal range		ritardo accesso cure
anemia		chemioprofilassi
hyperparasitemia		

Tabella 3.1: Elenco completo delle feature considerate per ogni paziente.

Prima di iniziare ad implementare i vari algoritmi di Machine Learning descritti nel Capitolo 2, è necessaria una fase di preparazione dei dati (**data preprocessing**) in modo da metterli nella forma corretta per lavorarci con gli strumenti a disposizione. In questo caso per l'analisi è stata utilizzata la libreria Python Scikit-Learn (https://scikit-learn.org/stable/). In questo ambiente, i dati sono organizzati in matrici, dove le righe rappresentano gli elementi del dataset mentre le colonne corrispondono alle feature. Tale **feature matrix** di dimensione $n \times k$, è convenzionalmente indicata con X. Le classificazioni corrette dei dati sono invece immagazzinate in un vettore y (**target vector**), di dimensione n (Figura 3.1). Nel caso in analisi, la generica coordinata j del vettore y è 0 se il j-esimo paziente è stato curato per via orale, mentre 1 se per via endovenosa.

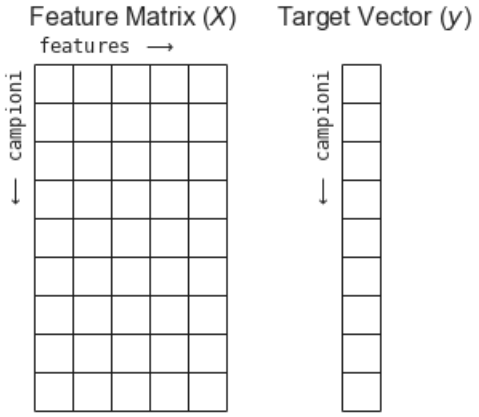


Figura 3.1: Rappresentazione dei dati in Scikit-Learn.

Quando si ha a che fare con l'analisi di dati reali, uno dei problemi che si incontra è la gestione dei dati mancanti (**missing values**) ovvero dei valori che, per qualche motivo, non è stato possibile recuperare. Per disporre della massima veridicità, si può scegliere di eliminare dal dataset i campioni con informazioni mancanti. Tuttavia, quando si dispone, come nel caso in questione, di un insieme di dati poco numeroso è sconsigliato sacrificare in questo modo dei campioni. Si preferisce invece inserire nelle entrate vuote la media dei valori relativi alla feature in questione, qualora essa sia numerica, e la moda nel caso di feature categoriche (cioè con valori nominali, come ad esempio la feature relativa alla zona dove è stata contratta la malaria).

Alcuni modelli, come random forest, si adattano sia a variabili numeriche che categoriche, altri invece possono operare unicamente con valori numerici. In questo caso, le feature categoriche devono essere convertite in numeriche facendo corrispondere ad ogni possibile valore nominale un numero naturale. Tuttavia questo approccio si rivela ben presto sconveniente per il fatto che i numeri possiedono un loro ordinamento che non trova un corrispettivo nei nomi, dando vita anche ad inconvenienti: nel caso della zona di infezione, assegnando ad esempio il valore 1 al Kenya, 2 all'Etiopia, 3 alla Nigeria, per alcuni algoritmi ciò implicherebbe Kenya $<$ Etiopia $<$ Nigeria.

Per questo motivo si utilizza la codifica **one-hot**, la quale consiste nell'aggiungere delle colonne alla feature matrix ognuna delle quali corrispondente ad una nuova variabile binaria per ogni possibile valore della feature categorica. Per l'esempio introdotto, si avrà:

$$\begin{bmatrix} Kenya \\ Etiopia \\ Kenya \\ Nigeria \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Infine, spesso è molto importante normalizzare i dati prima di utilizzarli. Il principale motivo è evitare che le feature che assumono valori numerici molto elevati risultino, per un algoritmo, più rilevanti delle feature con intervalli numerici più piccoli. Inoltre questo accorgimento permette anche di ridurre le difficoltà numeriche durante la fase di calcolo. Sia x^j il vettore n-dimensionale con i valori della j-esima feature. La formula che calcola la componente z_i^j del vettore normalizzato è la seguente:

$$z_i^j = \frac{x_i^j - \mu^j}{\sigma^j}, \quad i = 1, \ldots, n$$

dove μ^j e σ^j sono rispettivamente la media e la deviazione standard della j-esima feature.

Come alternativa alla normalizzazione, si può scegliere semplicemente di effettuare una riscalatura dei valori di ciascuna feature, per esempio riconducendosi all'intervallo $[-1; 1]$ o $[0; 1]$. Chiaramente bisogna usare lo stesso metodo di scalatura sia per il training set che per il testing set. Supponendo per esempio che la prima feature nel training set assuma valori in $[-10; 10]$ e venga scalata in $[-1; 1]$, allora, se la stessa feature nei campioni del testing set assume valori in $[-11; 8]$, questa dovrà essere scalata in $[-1, 1; 0, 8]$.

3.2 Costruzione dei classificatori SVM e Random Forest

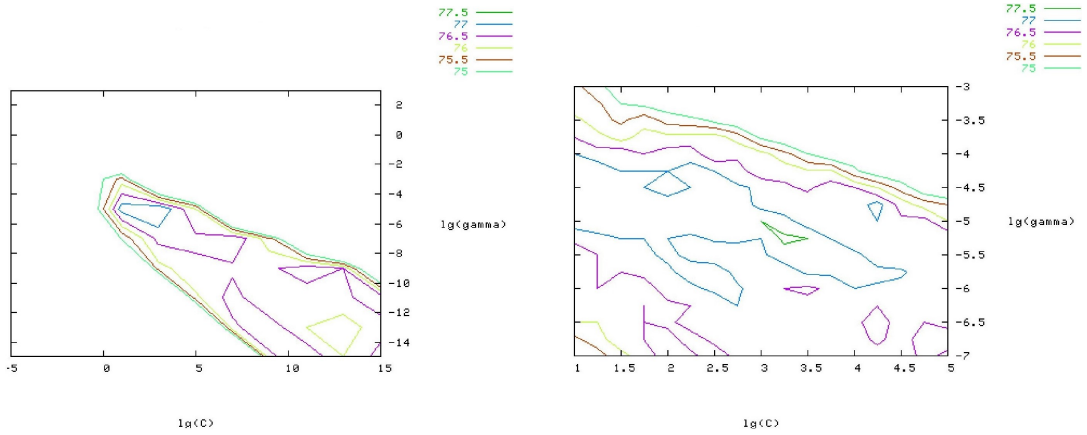
In generale la creazione ed applicazione di un modello con Scikit-Learn consta dei seguenti passaggi:

1. scelta di una classe di modelli e importazione dello stimatore desiderato;
2. eventuale specificazione degli iperparametri;
3. divisione dei dati in training set e testing set, e organizzazione degli stessi nella feature matrix e target vector, come discusso sopra;
4. addestramento del modello sul training set mediante il metodo fit();
5. applicazione del modello al testing set attraverso il metodo predict() [11].

Il primo strumento di apprendimento supervisionato impiegato come classificatore è stato SVM con kernel RBF. Come mostrato nella sezione 2.1, ci sono due iperparametri da scegliere per questo modello: C e γ. Per determinare il loro valore ottimale ci sono alcune strategie.

Grid search è tra quelle più tradizionali e consiste in una ricerca su una griglia bidimensionale, in combinazione a verifiche di accuratezza tramite cross validation. Come evidenziato in [15], una scelta efficace per i valori della griglia sono sequenze esponenzialmente crescenti di C e γ (per esempio $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}, \gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$).

Dal momento che eseguire una completa grid search può richiedere del tempo e un elevato costo computazionale, è consigliabile usare in un primo momento una griglia larga, grossolana, cioè in cui i valori di ciascun iperparametro siano ben distanziati tra loro, ed individuare una regione della griglia con buoni risultati di accuratezza, dove procedere con una griglia più fine (Figura 3.2).



(a) *Griglia larga* (b) *Griglia fine*

Figura 3.2: Esempio di ricerca degli iperparametri C e γ su una griglia larga e su una più fine.

Oltre a grid search, esistono metodi avanzati che garantiscono un minor costo computazionale. Tuttavia, avendo in questo caso solamente due iperparametri, il tempo richiesto da grid search per trovare dei buoni valori non è poi molto maggiore. Esiste inoltre una strategia ancora più semplice: la **random search**. Come suggerisce il nome, la ricerca dei valori ottimi degli iperparametri viene condotta su un insieme di valori casuali. Se questo garantisce un buon risparmio in termini di tempo, la mancata ricerca su una griglia ampia ed esaustiva non può chiaramente garantire l'ottimalità degli iperparametri individuati [16].

Senza specificare i valori di C e γ, l'accuratezza raggiunta è stata del $83,08\%$. Tale valore ha raggiunto l'$86,15\%$ inserendo i parametri ottimali selezionati dalla grid-search: $C = 5$ e $\gamma = 0,005$ [1].

[1] È stata usata la k-fold cross-validation con $k = 10$.

Il secondo modello usato per la costruzione di un classificatore è stato random forest. Gli iperparametri presi in considerazione in questo caso sono stati:

- `n_estimators`: specifica il numero di alberi nella foresta (il valore di default è 100). Aumentando questo parametro, si migliora la capacità di generalizzazione del modello ma allo stesso tempo si aumenta la complessità computazionale;
- `max_depth`: indica la massima profondità di ciascun albero. Se non viene specificato, ogni albero si espande finchè tutte le foglie diventano "pure", cioè contenenti campioni appartenenti alla stessa classe;
- `max_features`: è il massimo numero di feature considerate per lo split di ciascun nodo;
- `min_samples_split`: rappresenta il minimo numero di elementi per consentire uno split [17].

La grid-search convalidata con cross-validation ha individuato come ottimali 250 alberi con una profondità massima di 90. Il massimo numero di feature per split è stato lasciato al valore di default $\sqrt{k}$, così come il numero minimo di elementi per consentire uno split, cioè due. Con questi valori, l'accuratezza raggiunta è stata del $87,69\%$.

Questi risultati non eccellenti sono da attribuirsi principalmente a 2 cause: per prima cosa il database a disposizione ha un numero molto ristretto di campioni (119) per consentire una modellazione adeguata: possedendo una quantità maggiore di elementi, la fase di addestramento dei classificatori sarebbe stata più esaustiva, consentendo una descrizione più particolareggiata del fenomeno senza correre il rischio di cadere in overfitting e migliorando così la fase di generalizzazione. Secondariamente, non è da escludere la presenza di rumore dovuto a eventuali feature irrilevanti. Per questo, il passaggio successivo nell'analisi è stato quello di effettuare una feature selection allo scopo di ottimizzare le prestazioni di SVM e random forest.

3.3 Feature Selection

Per migliorare i classificatori, i due strumenti utilizzati per individuare le feature più rilevanti sono stati la selezione delle feature univariata (UFS) e random forest. Il primo è un metodo di tipo filtro e il suo funzionalmento consiste nell'esaminare ogni feature individualmente per determinare la relazione tra essa e l'output. La libreria Scikit-Learn fornisce la classe SelectKBest, abbinabile a svariati test statistici (ad esempio il test χ^2). Il valore scelto per quantificare la correlazione con l'output è l'ANOVA F-Value. Per quanto concerne random forest, durante la precedente fase di addestramento del modello, l'algoritmo stesso ha attribuito ad ogni feature un punteggio direttamente proporzionale alla sua importanza. Queste informazioni possono essere sfruttate, riconcependo random forest come un metodo di feature selection (di tipo embedded).

Le varie feature sono state riordinate in base a questi punteggi e i ranking ottenuti nei due casi sono illustrati nella Figura 3.3:

Feature Ranking tramite UFS	**Feature Ranking tramite Random Forest**
1. celebral malaria/coma (WHO)	1. bil 1° giorno
2. jaundice (WHO)	2 cerebral malaria/coma
3. hyperparasitemia (WHO)	3. parassitemia baseline %
4. parassitemia baseline %	4. PLT 1°giorno
5. anemia (WHO)	5. Hb 1° giorno
6. acute renal failure (WHO)	6. ALT 1° giorno
7. after 24 hr parassitemia baseline %	7. età
8. bil 1° giorno	8 after 24 hr parassitemia baseline %
9. creat 1° giorno	9. AST 1° giorno
10. AST 1° giorno	10. Na 1° giorno
11. shock (WHO)	11. durata permanenza (giorni)
12. comorbitiy	12. creat 1° giorno
13. hypoglycaemia (WHO)	13. comorbitiy
14. respiratory failure (WHO)	14. ritardo diagnosi (gg)
15. liver function test >3 time normal range (WHO)	15. ritardo accesso cure (gg)
16. PLT 1°giorno	16. anemia
17. ALT 1° giorno	17. country infection 1=West africa
18. chemoprophylaxis (atovaquone-proguanil)	18. chemoprophylaxis (Doxycycline)
19. acidosis (WHO)	19. jaundice
20. patient status (EU)	20. hyperparasitemia
21. convulsions (WHO)	21. sex
22. patient status (Imigrant)	22. patient status (EU)
23. spontaneous bleeding	23. patient status (Imigrant)
24. country infection 1=West africa	24. patient status (visitor from endemic country)
25. chemoprophylaxis (Mefloquine)	25. pregressa malaria

Figura 3.3: Le 25 feature più rilevanti secondo UFS e Random Forest.

Confrontando i risultati ottenuti, emergono alcune somiglianze e alcune interessanti discrepanze: entrambi i metodi evidenziano l'importanza del criterio WHO celebral malaria/coma, del livello di bilirubina (bil 1° giorno) e della presenza di parassiti nel sangue (parassitemia baseline % e after 24hr parassitemia baseline %). Inoltre, il ranking prodotto da UFS sottolinea come risultino rilevanti anche altri criteri WHO: oltre alla feature booleana (cioè che assume valore 1 o 0) celebral malaria/coma, il secondo e terzo posto di questa classifica sono occupati da jaundice (itterizia), parametro legato all'eccessivo innalzamento della bilirubina, e hyperparasitemia. Più in generale, 9 tra le prime 15 feature rientrano tra questi criteri. Random forest invece evidenzia anche il contributo di alcuni valori ematici: nelle prime posizioni si possono osservare l'indice PLT 1° giorno (che esprime il numero di piastrine per volume di sangue), il valore dell'emoglobina (Hb 1° giorno) e il livello degli enzimi transaminasi AST 1° giorno e ALT 1° giorno.

Una volta ricavate queste informazioni, sono stati ricostruiti i classificatori SVM e random forest su un dataset che tenesse conto solo delle feature più rilevanti. Non sapendo quante considerarne e nemmeno quale ranking fosse il migliore, sono state provate tutte le combinazioni possibili, procedendo nel modo seguente:

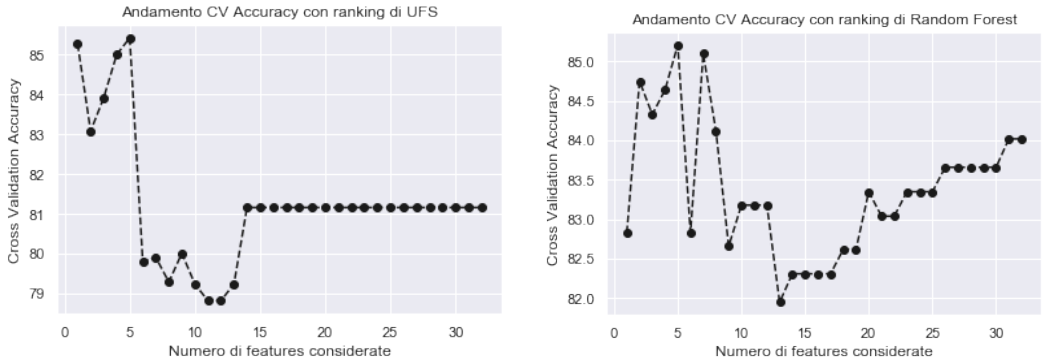
(i) scelta del ranking (UFS o random forest);

(ii) scelta del numero di feature (da 1 a 38);

(iii) eliminazione dalla feature matrix delle colonne relative alle feature trascurate;

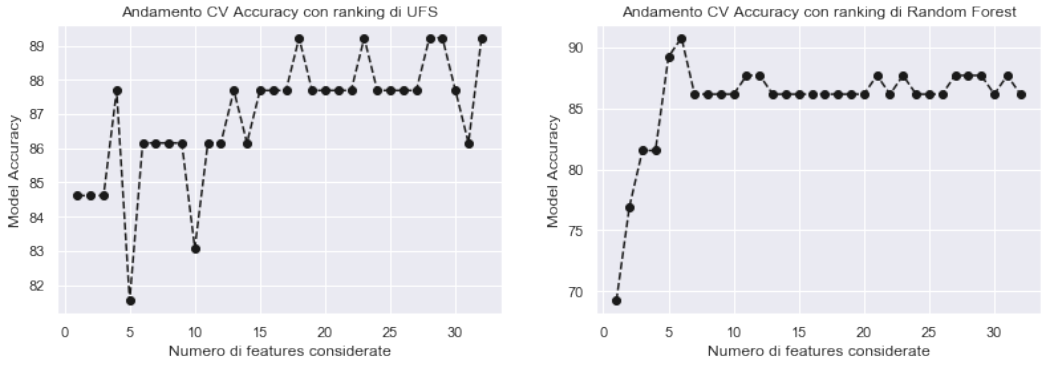
(iv) addestramento dei classificatori e calcolo della cross validation accuracy.

Questo procedimento è stato ripetuto per tutte le combinazioni possibili. Per completezza, vengono riportati i grafici dell'andamento dell'accuratezza dei modelli al variare del ranking e del numero di feature.



(a) *Ranking fornito da UFS* (b) *Ranking fornito da random forest*

Figura 3.4: Cross validation accuracy di SVM al variare del ranking e del numero di feature.



(a) *Ranking fornito da UFS* (b) *Ranking fornito da random forest*

Figura 3.5: Cross validation accuracy di Random Forest al variare del ranking e del numero di feature.

Fino a questo momento, non è stata effettuata alcuna selezione degli iperparametri, lasciandoli ai valori di default. L'ultimo passaggio quindi è stato quello di individuare le combinazioni che garantissero i risultati migliori, ed eseguire con queste una grid search. I risultati che terminano questa parte dell'analisi sono riportati nella seguente tabella:

Classificatore	**Strumento di Feature Ranking**	**Numero di Feature considerate**	**Valore degli Iperparametri**	**Cross validation accuracy**
SVM	UFS	7	C = 1 gamma = 0,25	92,31 %
SVM	Random Forest	7	C = 2 gamma = 0,0625	89,23 %
Random Forest	UFS	17	n_estimators = 200 max_depth = 50	89,23 %
Random Forest	Random Forest	5	n_estimators = 200 max_depth = 50	90,77 %

Tabella 3.2: Prestazioni dei classificatori dopo aver eseguito una Feature Selection.

Si può quindi osservare come l'aver considerato un numero minore di feature abbia effettivamente migliorato la precisione dei due classificatori: il valore della cross validation accuracy si aggira intorno al 90%, arrivando a superare il 92% nel caso di SVM con le prime 7 feature del ranking prodotto da UFS.

3.4 Clustering

Finora, per l'analisi del dataset sono state impiegate tecniche di apprendimento supervisionato, in quanto a tutti gli algoritmi implementati è stata fornita la corretta classificazione dei campioni durante la fase di addestramento. L'apprendimento consisteva dunque nel capire in che modo le varie feature dei campioni del training set portassero a quel determinato output (noto!) e, dopo aver approssimato questa relazione input-output, si sarebbero usate le informazioni ricavate per la fase di generalizzazione sul testing set.

In altri termini, i modelli precedenti avevano l'obiettivo di ricalcare il processo di decisione della terapia compiuto dai medici e, potenzialmente, di riuscire a sostituire il decisore. Nel fare ciò, è stata assunta l'assoluta correttezza delle scelte compiute nella realtà. L'aver ottenuto un buon livello di accuratezza, nonostante la limitata grandezza del dataset, può essere visto come una verifica della coerenza nel processo di decisione.

In questa sezione invece, viene descritto l'impiego di alcune tecniche di apprendimento non supervisionato. Lo scopo non è più quello di approssimare la funzione menzionata precedentemente, bensì quello di individuare una struttura intrinseca al dataset stesso. In particolare, sono stati usati alcuni algoritmi di clustering partizionale, i quali cercano di determinare delle naturali suddivisioni del dataset in vari gruppi, in base alla somiglianza (vicinanza) tra gli elementi.

Il principale vantaggio di questi strumenti sta nel fatto che, chiedendo una suddivisione in sole due parti del dataset, è possibile confrontare i risultati dell'algoritmo con quelli effettivi: ammesso che i dati in questione consentano una buona clusterizzazione, si potrà capire se alcuni dei pazienti che hanno affrontato la terapia endovenosa possano essere ricandidati buoni per la terapia orale o, viceversa, se per dei pazienti curati per via orale fosse stata preferibile la terapia più invasiva.

I due algoritmi di clustering considerati sono stati k-means e spectral clustering. È importante osservare che, se nella costruzione dei classificatori non era di fondamentale importanza normalizzare le varie feature, con questi algoritmi tale fase della preparazione del dataset risulta essere determinante, in quanto i valori numerici delle feature vengono utilizzati per calcolare delle distanze.

I primi risultati ottenuti si sono rivelati poco significativi, in quanto non si è evidenziata una relazione tra terapia effettuata e cluster di appartenenza: i pazienti con trattamento endovenoso sono stati ripartiti in modo quasi equo tra i due gruppi.

Le difficoltà riscontrare dai due algoritmi, sono state attribuite principalmente alla ridotta dimensione del dataset e alla possibile presenza di rumore. Ragionando come fatto precedentemente con i classificatori SVM e random forest, si è deciso di effettuare una feature selection e vedere se, al variare del numero di feature considerate, il clustering migliorasse. Non ottenendo questo effetto utilizzando il ranking prodotto dalla selezione delle feature univariata e quello di random forest, è stata introdotta l'analisi delle componenti principali (PCA). Come accennato nella sezione 2.3, questo strumento risulta utile in vari contesti: per esempio, trova applicazione in compiti di riduzione dimensionale, filtraggio del rumore e feature selection. Tuttavia, anche PCA nasconde alcune debolezze, in particolare tende ad essere fortemente influenzata da eventuali valori anomali (*outliers*) tra i dati. Per questa ragione, sono state sviluppate delle varianti tra cui **sparse PCA**, la quale calcola le componenti principali attraverso combinazioni lineari di un numero ristretto di variabili, invece che di tutte [18].

Il seguente grafico mostra quanto bene venga rappresentato il dataset in base al numero di features che si considerano: da esso si conclude che le 11 feature più rilevanti secondo PCA sono sufficienti a descrivere oltre il 90% della varianza del dataset.

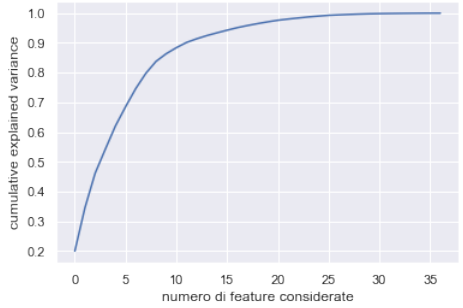


Figura 3.6: Varianza del dataset descritta in funzione del numero di feature considerate.

Vengono riportati i risultati ottenuti con le diverse combinazioni tra i due algoritmi di clustering e i due strumenti di feature selection:

Algoritmo di Clustering	K-Means	Spectral Clustering	K-Means	Spectral Clustering
Strumento di Feature Selection	PCA	PCA	sPCA	sPCA
Numero di pazienti nel cluster più grande	95	67	114	74
di cui con terapia orale	60	44	65	46
Numero di pazienti nel cluster più piccolo	24	52	5	45
di cui con terapia orale	9	25	4	23

Tabella 3.3: Risultati del Clustering preceduto da una feature selection.

Purtroppo, neppure la fase di feature selection ha reso più semplice il compito degli algoritmi di cluster: escludendo il risultato ottenuto da k-means con sPCA, nessuna combinazione di quelle provate è riuscita a raccogliere più del 70% di pazienti con terapia endovenosa in uno dei due cluster.

Capitolo 4

Conclusioni

Il primo risultato utile dello studio compiuto è stata la costruzione dei classificatori SVM e random forest. Dopo averne migliorato le prestazioni attraverso una feature selection e una ricerca degli iperparametri ottimali, sono stati raggiunti livelli di accuratezza abbastanza elevati (tra l'89% e il 93%), valori che rendono questi modelli degli strumenti utili e abbastanza precisi per la selezione della terapia nel caso di malaria severa.

Il feature ranking ottenuto attraverso la selezione univariata delle feature e random forest, ha permesso di individuare i parametri clinici che risultano essere maggiormente decisivi nell'optare o meno per il trattamento endovenoso. Per prima cosa è stata confermata l'importanza degli attuali criteri WHO, in particolare celebral malaria/coma, jaundice e hyperparasitemia. Oltre ad essi, sono stati individuati come segnali rilevanti anche il livello di parassiti nel sangue (parassitemia baseline % e after 24hr parassitemia baseline) e altri valori ematici (bil, PLT e Hb).

A dimostrazione della coerenza dei risultati ottenuti, si può osservare come alcuni di questi parametri siano effettivamente legati. Per esempio, la bilirubina (bil) è una sostanza che deriva dalla degradazione dell'emoglobina (Hb) e normalmente la sua produzione è bilanciata da un meccanismo di eliminazione. Il mancato raggiungimento di tale equilibrio, con l'innalazamento del livello di bilirubina, porta alla condizione clinica di itterizia (jaundice).

Si osservi che mentre i criteri WHO sono feature booleane, e quindi i casi gravi possono essere identificati dalla realizzazione o meno di uno di tali criteri, le altre feature sono variabili continue a valori in intervalli numerici, perciò risulta utile considerare dei valori soglia per tali parametri (il dataset fornito aveva già indicato alcuni valori limite assunti attualmente dai medici).

Bibliografia

[1] O. Cominetti, *Identifcation of a Novel Clinical Phenotype of Severe Malaria using a Network-Based Clustering Approach*, in AA.VV., in nature.com/scientificreports, 27 Agosto 2018.

[2] *Malaria - Aspetti epidemiologici*, in D. Boccolini, C. Severini e L. Gradoni (a cura di), in epicentro.iss.it/malaria/epidemiologia-europa, 1 agosto 2019.

[3] L. Grippo, M. Sciandrone, *Metodi di ottimizzazione per le reti neurali*, pp.11-12, 17, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 2003.

[4] F. Scala, *Machine Learning - una introduzione dettagliata*, 2018.

[5] J. Chapmann, *Neural Networks: Introduction to Artificial Neurons, Backpropagation Algorithms and Multilayer Feedforward Neural Networks*, vol 2, 2017.

[6] J. Brownlee, *An Introduction to Feature Selection*, in machinelearningmastery.com, 6 Ottobre 2014.

[7] V. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[8] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.

[9] O. L. Mangasarian, *Nonlinear Programming*, SIAM, Philadelphia, 1994.

[10] L. Grippo, M. Sciandrone, *Metodi di ottimizzazione per le reti neurali*, pp 59-62, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 2003.

[11] J. VanderPlasp, *The Python Data Science Handbook*, O'Reilly, 2016.

[12] S. Hartshorn, *Machine Learning With Random Forests And Decision Trees*, Seattle, 2016.

[13] N. Liberman, *Decision Trees and Random Forests*, in towardsdatascience.com, 27 Gennaio 2017.

[14] W. Fleshman, *Spectral Clustering - Foundation and Application*, in towardsdatascience.com, 21 Febbraio 2019.

[15] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, *A Practical guide to Support Vector Classication*, Department of Computer Science National Taiwan University, Taiwan, 2016.

[16] T. Boyle, *Hyperparameter Tuning*, in towardsdatascience.com, 16 Febbraio 2019.

[17] R. Meinert, *Optimizing Hyperparameters in Random Forest Classification*, in towardsdatascience.com, 5 Giugno 2019.

[18] Ganesh R. Naik, *Advances in Principal Component Analysis: Research and Development*, Springer, 2018.

[19] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, 2011.