Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Bioingegneria

Classificazione dello stato di avanzamento del diabete tramite indici di variabilità glicemica

Laureando

Enrico Longato

Relatore

Giovanni Sparacino

Co-relatore

Andrea Facchinetti

Anno Accademico 2015/2016

Abstract

Contestualmente allo sviluppo della tecnologia per il monitoraggio in continua del glucosio (Continuous Glucose Monitoring, CGM), è emersa l'opportunità di caratterizzare quantitativamente l'andamento della glicemia sulla base delle sue proprietà dinamiche. Proprio a questo scopo, sono stati proposti in letteratura svariati indici di variabilità glicemica (Glycemic Variability, GV) il cui numero si attesta, ormai, sulle diverse decine. Molti di questi indici si sono dimostrati sensibili ad alterazioni del metabolismo, al diabete e, in particolare, al suo stato di avanzamento. Nonostante questo, nessuno di essi, da solo o combinato con altri, è emerso come gold standard da affiancare alla misurazione di laboratorio dell'emoglobina glicata (HbA1c) o all'invasiva OGTT (Oral Glucose Tolerance Test) per la diagnosi del diabete di tipo 2 (T2D) o di stati metaboliche ad esso correlati come l'Impaired Glucose Tolerance (IGT).
In questo contesto, la messa a punto di un sistema diagnostico efficace basato sull'analisi del segnale CGM (equiparabile, in questo senso, ad un Holter per la glicemia) si configura come un obiettivo fortemente desiderabile e dalle notevoli ramificazioni in ambito clinico ed economico. Infatti, innanzi tutto, la minima invasività e la facilità di mantenimento dei sensori CGM garantirebbero una notevole riduzione del disagio a carico del paziente e dei costi, diretti e sociali, correlati alla necessità di sottoporsi a più lunghe e complesse analisi di laboratorio. Inoltre, allo stato attuale, una diagnosi basata sull'HbA1c permette di ottenere soltanto un quadro riassuntivo della condizione metabolica media del soggetto riferito all'arco di alcuni mesi (tempo di vita dei globuli rossi). Al contrario, un'opportuna elaborazione dei dati CGM potrebbe produrre risultati con risoluzione temporale molto fine, nell'ordine di circa una settimana. In questo modo, sarebbe possibile monitorare quasi costantemente lo stato di salute del paziente, riducendo le visite inopportune dallo specialista e aumentando la tempestività nell'adozione di misure terapeutiche correttive.
Con questo lavoro, dunque, ci si è proposti di dimostrare che opportune combinazioni di indici di GV estraibili da segnali CGM, completate da alcune informazioni cliniche di base (quali età, sesso, BMI e circonferenza della vita), permettono, ricorrendo a tecniche di classificazione opportune, di differenziare con accuratezza tra pazienti IGT e T2D anche senza avere a disposizione grandi moli di dati o gli esiti di test clinici ad hoc come l'OGTT. A tale scopo, attraverso il progetto di ricerca FP7 "Mosaic" di cui l'Università di Padova è partner, si è utilizzato un database relativo a 62 pazienti, per ciascuno dei quali si hanno a disposizione tracciati CGM dalla durata media di circa una settimana e le altre informazioni cliniche di base raccolte in concomitanza di due visite a cadenza annuale. Nello specifico, dopo aver approntato tre scenari di classificazione di complessità e contenuto informativo crescenti, si è partiti con lo stabilire una baseline di accuratezza tramite tre tecniche molto semplici: un classificatore lineare con indicator matrix coding, il Fisher's Linear Discriminant e la regressione logistica. In tutti i casi, le prestazioni si sono rivelate non particolarmente elevate (accuratezza circa del 70%). Risultati migliori (accuratezza superiore all'80%) si sono ottenuti, invece, passando al classificatore K-Nearest Neighbours, una tecnica non parametrica che risulta, però, di difficile scalabilità. Successivamente, nel tentativo di migliorare ulteriormente le performance

di classificazione, si sono intraprese le due strade, concettualmente complementari, dell'applicazione dei metodi di ensemble e dell'aumento della complessità dei classificatori impiegati. Seguendo quest'ultimo filone logico, si è individuata la Support Vector Machine con kernel polinomiale, applicata agli indici di GV e ad alcuni parametri clinici di base (età, sesso, BMI e circonferenza della vita), come la tecnica più adatta alla distinzione tra IGT e T2D (accuratezza pari all'87.1%).
Infine, con la speranza di superare alcune difficoltà interpretative correlate all'utilizzo del kernel trick, si è approntato un procedimento di feature selection manuale grazie al supporto dato da un esperto diabetologo, incaricato di escludere gli indici o i parametri da lui considerati meno attendibili. In questo modo, si è potuto risolvere il problema di classificazione tramite una semplice SVM lineare, peggiorando solo marginalmente le prestazioni (accuratezza dell'85.5%).
Possibili sviluppi futuri includono l'estensione a popolazioni più numerose con l'obiettivo di generalizzare i risultati ottenuti e la messa a punto di un ulteriore sistema di monitoraggio che permetta di seguire l'evoluzione dello stato metabolico del paziente con frequenza molto più fine di quella permessa dalla sola HbA1c.

Indice

Capitolo 1

Il concetto di variabilità glicemica

1.1 Premessa

In questo capitolo introduttivo si delineano alcuni concetti fondamentali circa la patologia del diabete e la misurazione della glicemia. L'obiettivo è quello di presentare una panoramica sintetica dell'argomento, mettendo in evidenza gli aspetti critici relativi alla gestione della malattia che, in ultima analisi, hanno spinto all'adozione di strumenti per il monitoraggio in continua del glucosio e all'elaborazione di alcuni indici riassuntivi dello stato metabolico del paziente.
Per questo motivo, nel seguito, non ci si soffermerà approfonditamente su tutti gli aspetti clinici e fisiologici, ma si privilegeranno quelli maggiormente rappresentativi di ipotesi o condizioni al contorno necessarie per la comprensione e l'analisi del problema dal punto di vista modellistico.

1.2 Il diabete

Il diabete mellito è una patologia metabolica caratterizzata da alterazioni dei meccanismi di secrezione o azione dell'insulina, un ormone il cui compito principale è quello di permettere il passaggio dal sangue alle cellule del glucosio, loro principale fonte di energia [1].
Esistono due classi cliniche principali di diabete: il diabete di tipo 1, altresì detto insulino-dipendente, e il diabete di tipo 2, noto anche come insulino-resistente.
In generale, il diabete di tipo 1 insorge nei primi anni di vita e si aggrava molto rapidamente. Esso è causato dalla scarsità (o assenza) di cellule β nel pancreas, il che determina una secrezione insufficiente di insulina che deve essere, quindi, introdotta in circolo dall'esterno. Diversamente, il diabete di tipo 2 si sviluppa molto più lentamente ed è tipico di soggetti obesi o anziani. Inoltre, poiché presenta sintomi più lievi e che si manifestano gradatamente, può passare inosservato per lunghi periodi. Sotto l'etichetta di diabete di tipo 2, in realtà, si fa riferimento a un gruppo di malattie in cui l'attività di regolazione ad opera dell'insulina è alterata: da qualche parte nel sistema di controllo della glicemia c'è un difetto tale per cui, sebbene l'insulina venga prodotta, essa non sortisce

l'effetto desiderato. In entrambi i casi, il paziente deve bilanciare attentamente l'assunzione di carboidrati e le iniezioni di insulina in modo da mantenersi nella condizione favorevole di euglicemia.
Sebbene per i soggetti sani la concentrazione del glucosio nel sangue ci si aspetta debba variare tra 80 e 120 mg/dl [2], la fascia euglicemica si definisce come l'intervallo $70 \div 180$ mg/dl entro cui è più realistico che un paziente diabetico possa mantenersi [3].
Qualora la concentrazione del glucosio nel sangue ecceda i 180 mg/dl si parla di iperglicemia. La permanenza in tale condizione per lunghi intervalli di tempo può dar luogo, nel medio e lungo termine, a complicanze a carico, per esempio, del sistema nervoso e cardiovascolare, quali retinopatie (che possono portare a cecità irreversibile) e neuropatie. Situazione ben più grave, specie nel breve periodo, è rappresentata, invece, da ricorrenti episodi di ipoglicemia, cioè tali per cui la concentrazione del glucosio nel sengue sia inferiore ai 70 mg/dl. Infatti, già permanenze molto brevi in lieve ipoglicemia possono dare luogo a sonnolenza e perdita delle forze, il che può essere causa di incidenti che possono coinvolgere il paziente stesso, magari in seguito a una caduta, ma anche chi gli sta intorno, qualora si trovi alla guida o in situazioni simili. Casi di ipoglicemia grave e prolungata, invece, possono portare al coma o, addirittura, alla morte e sono comunque correlati a un innalzamento del tasso di mortalità. Per queste ragioni, l'obiettivo generale in ambito clinico è quello di limitare il più possibile l'ipoglicemia, accettando, invece, alcuni episodi di iperglicemia [4].

1.3 Monitoraggio in continua del glucosio: CGM

Il monitoraggio della glicemia con fini di controllo viene tradizionalmente effettuato tramite un protocollo, piuttosto laborioso per il paziente, basato sulla frequente valutazione della concentrazione di glucosio nel sangue tramite appositi dispositivi [5]. Questo procedimento, che comporta un disagio non indifferente per il soggetto, è noto, in gergo, con l'acronimo SMBG (*Self-Monitoring of Blood Glucose*). La principale limitazione di questo approccio, che, pure, mette a disposizione dati molto precisi, è proprio la saltuarietà del campionamento che rende difficile, per alcuni pazienti, un adeguato controllo della glicemia.
Il vantaggio che una tecnologia atta a produrre un profilo glicemico sostanzialmente continuo offre in tal senso è abbastanza evidente. Proprio per questo, in tempi recenti, si è assistito alla diffusione e allo sviluppo di dispositivi e tecniche per il monitoraggio in continua del glucosio (CGM, *Continuous Glucose Monitoring*). Tuttavia, per quanto l'adozione nella pratica clinica di routine del CGM sia una prospettiva allettante, si possono individuare alcuni punti critici [6]:

- mantenere un canale di comunicazione diretta tra lo strumento di misura e il flusso sanguigno per lunghi periodi è complicato e disagevole per il paziente;
- la precisione e le performance dei dispositivi sono variabili e devono documentate accuratamente prima di un utilizzo estensivo;
- come per tutte le nuove tecnologie, devono essere prese in considerazione le conseguenze socioeconomiche dell'adozione di massa del CGM.

Tralasciando quest'ultimo punto, poco pertinente e di difficile discussione, si possono fare, invece, alcune considerazioni tecnologiche di massima circa l'implementazione dei sensori CGM.
La quasi totalità degli strumenti per il monitoraggio in continua del glucosio utilizzati correntemente sui pazienti sono sensori elettrochimici transcutanei [7] composti, in prima approssimazione, da tre elementi principali [8]:

- un ago inserito a livello dell'addome (o anche, più raramente, della schiena o delle natiche) attraverso cui un filo viene posto a contatto con il liquido interstiziale e ne misura elettrochimicamente la concentrazione di glucosio;
- un trasmettitore wireless;
- una combinazione di ricevitore e monitor che permette al paziente di visualizzare il tracciato in tempo reale.

In particolare, nelle applicazioni di maggior successo il *sensing element* vero e proprio è un elettrodo sul quale siano stati opportunamente fissati appropriati enzimi. Si tratta, inoltre, quasi sempre, di sensori amperometrici [9]. Questo significa che la corrente misurata all'elettrodo è correlata alla concentrazione di glucosio presente nell'interstizio che, a sua volta, è nota essere un'ottima rappresentazione della glicemia vera e propria [10]. Si possono così ottenere misure stabili a distanza di appena qualche minuto l'una dall'altra; quindi a una frequenza molto elevata rispetto alle fluttuazioni significative del segnale. Il risultato è un tracciato sostanzialmente continuo che permette di visualizzare e studiare la variazione temporale della glicemia.
Volendo, infine, dare una misura indicativa della performance di questo genere di dispositivi, in letteratura si registrano valori di MARD (media delle differenze assolute espressa come percentuale del valor medio) tra segnale CGM e letture SMBG intorno al $10\% \div 15\%$, a seconda degli algoritmi correttivi utilizzati, con un certo peggioramento per campioni in fascia ipoglicemica.[11][12][13].

1.4 Diagnosi del diabete

La diagnosi del diabete può essere effettuata in maniera univoca servendosi di diverse tecniche, tutte egualmente degne di essere definite *gold standard*. In particolare, secondo la *best practice* individuata dalla American Diabetes Association [3], un soggetto è affetto da diabete se si verifica almeno una tra le seguenti situazioni:

1. emoglobina glicata (HbA1c) almeno pari al 6.5%, misurata in laboratorio tramite test accreditato;
2. glucosio plasmatico a digiuno (FPG, *Fasting Plasma Glucose*) almeno pari a 126 mg/dl dopo otto ore di digiuno;
3. glucosio plasmatico a due ore dall'inizio dell'OGTT (*Oral Glucose Tolerance Test*) almeno pari a 200 mg/dl;
4. misurazione casuale di glucosio superiore a 200 mg/dl in concomitanza con i tipici sintomi associati all'iperglicemia.

Qualora dovessero esserci risultati discordanti tra i test o il personale clinico non dovesse essere convinto circa la veridicità dell'esito dell'esame poiché, per esempio, non compatibile con i sintomi esibiti dal paziente, la raccomandazione è quella di ripetere le analisi.
Sulla base delle specifiche sopra indicate, inoltre, si possono individuare due categorie (a intersezione non vuota) di pre-diabete:

- IGT (*Impaired Glucose Tolerance*), caratterizzata da glucosio a due ore dall'OGTT tra 140 e 199 mg/dl;
- IFG (*Impaired Fasting Glucose*), caratterizzata da FPG tra 100 e 125 mg/dl.

Si tratta, nello specifico, di due patologie metaboliche che, pur non essendo definibili come diabete di tipo 2 a tutti gli effetti, ne aumentano il rischio e ne condividono alcune caratteristiche.

1.5 Variabilità glicemica: GV

Non c'è alcun dubbio all'interno della comunità scientifica che approntare una terapia con lo scopo di diminuire il livello di HbA1c nel paziente diabetico possa portare a risultati favorevoli nella riduzione delle complicanze a carico del sistema cardiocircolatorio. Tuttavia, alcuni trial clinici hanno evidenziato come questo approccio a volte possa non essere sufficiente.
A fronte di questa apparente incongruenza, si è aperto un dibattito con lo scopo di individuare il modo migliore per porre rimedio al problema [14][15]. L'attenzione si è spostata, quindi, verso la ricerca di metriche complementari all'emoglobina glicata che, nonostante sia un parametro irrinunciabile per la diagnosi del diabete e del rischio a esso correlato, è caratterizzata da una risoluzione temporale potenzialmente insoddisfacente. Infatti, per sua natura, essa cattura soltanto le lentissime variazioni nella concentrazione media di glucosio nel sangue che si registrano nell'arco di alcuni mesi [16].
È proprio qui che entra il gioco il concetto di variabilità glicemica (GV, *Glycemic Variability*) che fa riferimento, in generale, alle fluttuazioni abbastanza rapide della glicemia tra i suoi picchi e nadir.
Per lungo tempo, il maggior ostacolo all'adozione della GV come metrica attendibile è stata, semplicemente, la difficoltà di misurarla in maniera attendibile [17], ma il perfezionamento della tecnologia CGM presentata nella sezione 1.3 sembra poter permettere grandi progressi in questo senso. Tuttavia, come sarà evidente nel capitolo successivo, non si può ancora parlare di una standardizzazione delle metriche di GV che, anzi, sono proliferate in maniera quasi incontrollata. In ogni caso, prima di scendere nei dettagli matematici di ogni singolo indice, è bene evidenziare come ciascuno di essi metta in luce una o entrambe le seguenti caratteristiche del segnale CGM:

- l'ampiezza delle fluttuazioni, che indica la differenza di concentrazione tra due punti del tracciato senza, però, metterli in relazione temporale tra loro;
- il tempismo delle fluttuazioni, che evidenzia la durata di alcuni eventi significativi come, per esempio, la permanenza in ipoglicemia.

Capitolo 2

Indici di GV, uso clinico e scopo della tesi

2.1 Premessa

In questo capitolo si presentano tutti gli indici di variabilità glicemica che sono stati considerati all'interno del lavoro, corredandoli, qualora pertinente, di alcune note implementative che si pensa possano essere utili in vista di una futura discussione sulla standardizzazione delle tecniche di calcolo. Per tentare di fornire un'organizzazione logica degli indici in questione, essi vengono suddivisi tra le varie sezioni a seconda delle caratteristiche che li accomunano.
Resta inteso che le definizioni sono riferite al caso di dati estratti da un sensore CGM ed espressi con unità di misura mg/dl (conversione: 1 mmol/l $\simeq$ 18 mg/dl). Alcuni simboli ricorrenti sono:

- $G(t)$: valore che il tracciato CGM assume all'istante t;
- $G(t_k)$: valore che il tracciato CGM assume all'istante t_k, in concomitanza del quale è stato acquisito il k-esimo campione in esame;
- $G(k)$: valore che il tracciato CGM assume all'istante corrispondente al k-esimo campione in esame;
- N: numero totale di campioni a disposizione;
- N_{days}: numero di giorni per cui il segnale CGM è stato acquisito;

2.2 Indici basati sulle proprietà statistiche

Questa prima sezione raccoglie gli indici che catturano le proprietà statistiche del tracciato CGM [16][18]. Nonostante siano molto semplici, portano, in genere, delle informazioni molto significative e particolarmente apprezzabili in ambito clinico poiché facilmente interpetabili.
I valori degli indici sono riportati, qualora pertinente, nelle figure 2.1 e 2.2.

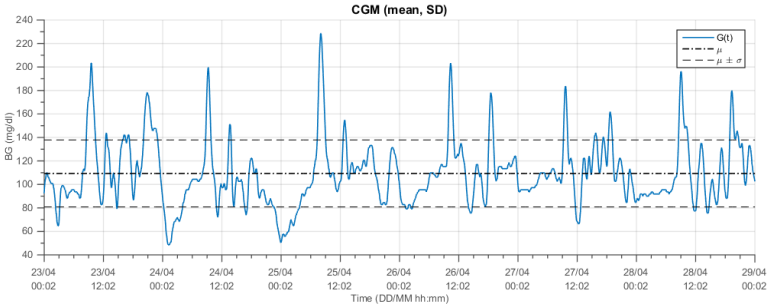


Figura 2.1: Media e deviazione standard del tracciato CGM. Le linee tratteggiate delimitano la fascia larga $2 \cdot \sigma_G$ e centrata in μ_G. Inoltre, il J-index corrisponde, a meno di un fattore di scala, al quadrato del valore della riga superiore.

2.2.1 Media

Il livello medio del glucosio su tutto il tracciato CGM a disposizione si calcola tramite la semplice media artimetica sui campioni:

$$m_G \coloneqq \frac{1}{N} \sum_{k=1}^{N} G(k) \tag{2.1}$$

2.2.2 Deviazione standard (SD)

La deviazione standard è una misura della variabilità del segnale CGM attorno alla media m_G. Uno stimatore non polarizzato della deviazione standard è

$$\sigma_G \coloneqq \sqrt{\frac{1}{N-1} \sum_{k=1}^{N} [G(k) - m_G]^2} \tag{2.2}$$

2.2.3 Coefficiente di variazione (CV)

Il coefficiente di variazione definisce una relazione tra la media e la deviazione standard, evidenziandone la dimensione relativa. Si esprime, spesso, in percentuale come

$$CV_{\%} \coloneqq 100 \cdot \frac{\sigma_G}{m_G} \tag{2.3}$$

2.2.4 Mediana

La mediana è il valore corrispondente al cinquantesimo percentile di un insieme ordinabile. In pratica, si calcola ordinando un vettore di dati e considerando quello che cade esattamente nel mezzo. Nel caso vi siano un numero pari di elementi, è la media aritmetica tra i due valori centrali.
Formalizzando, definito G_{sort} il segnale contenente tutti i campioni $G(k)$ ordinati in senso crescente o decrescente, si può scrivere

$$m_{50\%} \coloneqq \begin{cases} G_{sort}\left(\dfrac{N}{2}\right) & \text{, se N è pari} \\ G_{sort}\left(\dfrac{\lfloor N/2 \rfloor + \lceil N/2 \rceil}{2}\right) & \text{, se N è dispari} \end{cases} \tag{2.4}$$

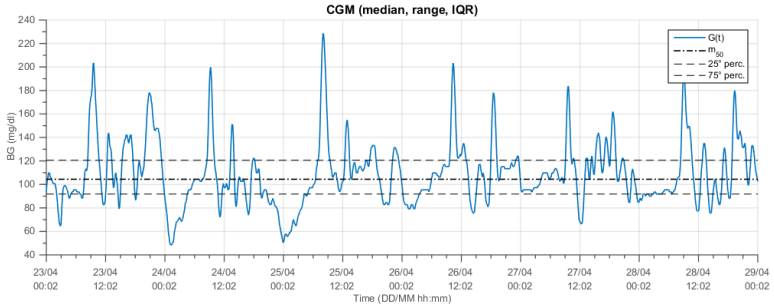


Figura 2.2: Mediana e IQR del tracciato CGM. Le linee tratteggiate delimitano la fascia compresa tra il 25° e 75° percentile.

2.2.5 Range

Il range è la differenza tra il valore massimo e il valore minimo di un insieme ordinabile:

$$\Delta G_r := \max_k[G(k)] - \min_k[G(k)] \tag{2.5}$$

2.2.6 Range inter-quartile (IQR)

Il range inter-quartile è la differenza tra il settantacinquesimo e venticinquesimo percentile di un insieme ordinabile.
Definito G^+_{sort} l'insieme di tutti i campioni $G(k)$ ordinati in senso crescente, si può scrivere

$$\text{IQR} := G^+_{sort}(0.75 \cdot N) - G^+_{sort}(0.25 \cdot N) \tag{2.6}$$

dove, per brevità, si è trascurato il caso, di facile risoluzione, in cui non ci sia un'esatta corrispondenza tra indice e campione. Rispetto al semplice range o alla deviazione standard, l'IQR è più adatto all'analisi del tracciato CGM a causa della non simmetria della scala glicemica rispetto al valor medio.

2.2.7 J-index

Il J-index [19] è una combinazione non lineare di media e varianza che mira a sintetizzare in un'unica variabile il contenuto informativo del vettore bidimensionale $(m_G \ \sigma_G)^\mathsf{T}$. Si definisce come:

$$J := \frac{(m_G + \sigma_G)^2}{1000} \tag{2.7}$$

2.3 Indici basati sulla variazione delle proprietà statistiche

Gli indici presentati fino ad ora sono indicativi rispetto alle proprietà statistiche del tracciato CGM visto in un'ottica statica. In altre parole, non si tiene conto delle loro variazioni temporali. Per tentare di colmare questa lacuna, sono stati proposti i due indici SD_w e SD_m [20].

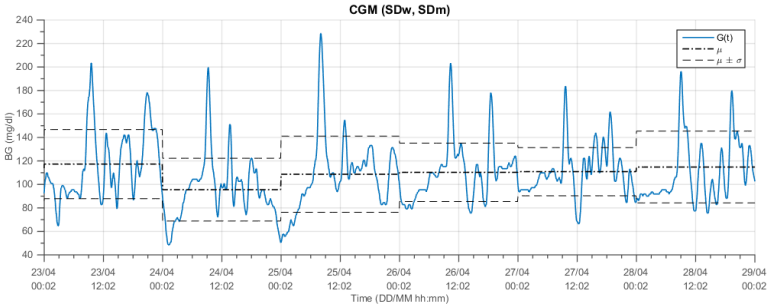


Figura 2.3: Medie e deviazioni standard giornaliere del tracciato CGM. Le linee tratteggiate delimitano la fascia larga $2 \cdot \sigma_G$ e centrata in μ_G relativa a ogni giornata.

Il primo si calcola come la media delle deviazioni standard giornaliere $\sigma_G(d)$

$$\mathrm{SD}_w \coloneqq \frac{1}{N_{days}} \sum_{d=1}^{N_{days}} \sigma_G(d) \tag{2.8}$$

Il secondo, in maniera speculare, rappresenta la deviazione standard delle medie giornaliere $m_G(d)$

$$\mathrm{SD}_m \coloneqq \sqrt{\frac{1}{N_{days}-1} \sum_{d=1}^{N} [m_G(d) - \mu_G]^2} \tag{2.9}$$

con

$$\mu_G \coloneqq \frac{1}{N_{days}} \sum_{d=1}^{N_{days}} m_G(d) \tag{2.10}$$

che coincide con m_G se ogni giorno è descritto dallo stesso numero di campioni. La variabilità della medie e delle deviazioni standard giornaliere è messa in evidenza in figura 2.3.

2.4 Indici basati sulla permanenza in un *target range*

Un indicatore semplice e diretto del controllo glicemico è dato dal tempo trascorso entro, sopra e sotto il *target range* $[70, 180]$ mg/dl, rappresentativo della condizione di euglicemia [18].
Si definiscono, quindi, tre indici percentuali:

$$\begin{aligned} \%\mathrm{BG}_{within} &\coloneqq 100 \cdot \frac{N_{eu}}{N} \\ \%\mathrm{BG}_{above} &\coloneqq 100 \cdot \frac{N_{hyper}}{N} \\ \%\mathrm{BG}_{below} &\coloneqq 100 \cdot \frac{N_{hypo}}{N} \end{aligned} \tag{2.11}$$

dove N_{eu}, N_{hyper}, N_{hypo} sono il numero di campioni in ciascuna delle tre fasce. Da come gli indici sono formulati traspaiono, evidentemente, alcune proprietà:

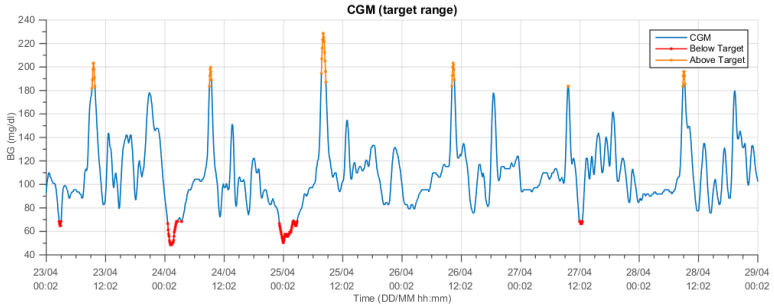


Figura 2.4: Target range del tracciato CGM. Il segnale è colorato in modo da evidenziare la fascia a cui appartiene ciascun campione: azzurro per l'euglicemia, arancio per l'iperglicemia e rosso per l'ipoglicemia

1. sommano a 100%;

2. non tengono conto dell'ampiezza del segnale;

3. sono riconducibili a una diretta intepretazione temporale solo in caso di tempi di acquisizione abbastanza lunghi (almeno una giornata) e tempo di campionamento circa uniforme.

2.5 Indici basati sul confronto con il passato

Per catturare, in qualche modo, la variabilità giornaliera del tracciato CGM, sono stati proposti in letteratura due indici abbastanza simili, CONGA e MODD, che si basano sulla differenza tra coppie di campioni separati da un intervallo di tempo fissato. [21]
Si fa presente che le semplici formule presentate di seguito non tengono conto del problema pratico dell'asincronia all'interno del segnale. In altre parole, nessuno garantisce che, fissato un intervallo di tempo costante ΔT, esistano contemporaneamente i campioni $G(t)$ e $G(t - \Delta T)$. Tuttavia, poiché il segnale CGM è piuttosto regolare, si propone di ovviare al problema interpolandolo linearmente (cioè congiungendo con una retta tutte le coppie di punti adiacenti) e ricampionandolo in modo tale da ripristinare la sincronia e poter applicare tranquillamente le formule. Si tratta, chiaramente, di una scelta arbitraria, ma apparentemente ragionevole proprio perché $G(t)$ è particolarmente *smooth*. In prima approssimazione, infatti, sotto l'ipotesi che il tempo di campionamento iniziale T_s sia sufficiente a catturare tutte le variazioni significative del segnale, ricampionare a $T_v < T_s$ in seguito a un'interpolazione lineare, fa commettere un errore nell'ordine di

$$\varepsilon \approx \frac{T_v}{T_s} \cdot |G(k) - G(k - T_s)|$$

il che sembra più che accettabile.
Si fa presente, inoltre, che per entrambi è indifferente considerare la differenza tra presente e passato o futuro e presente.

2.5.1 Continuous Overall Net Glycemic Action (CONGA)

L'indice CONGA si calcola come la deviazione standard tra il valore assunto da $G(t)$ a ogni istante e un certo numero di ore prima:

$$\mathrm{CONGA}_H \coloneqq \sqrt{\frac{1}{N_v - k_v^*} \sum_{k_v=k_v^*}^{N_v} [G(k_v) - G(k_v - H_v)]} \tag{2.12}$$

dove k_v^* indica il primo istante per cui entrambi i termini della differenza sono definiti, N_v è il numero di campioni k_v ottenuti tramite ricampionamento e H_v rappresenta lo scarto in campioni corrispondente a H (di solito 4) ore.

2.5.2 Mean of Daily Differences (MODD)

L'indice MODD, invece, è la media delle differenze assolute tra il valore di $G(t)$ a ogni istante e il suo corrispondente un giorno in anticipo:

$$\mathrm{MODD} \coloneqq \frac{1}{N_v - k_v^* + 1} \sum_{k_v=k_v^*}^{N_v} |G(k_v) - G(k_v - \Delta T_v)| \tag{2.13}$$

dove k_v^* indica il primo istante per cui entrambi i termini della differenza sono definiti, N_v è il numero di campioni k_v ottenuti tramite ricampionamento e ΔT_v rappresenta lo scarto in campioni corrispondente a 24 ore.

2.6 Indici indicativi di escursioni significative

Da un'analisi, anche cursoria, del segnale CGM si può osservare come le sue caratteristiche principali siano desumibili dalla frequenza e dall'entità dei picchi o, in generale, delle inversioni di tendenza. Attraverso gli indici presentati in questa sezione ci si propone, quindi, di catturare proprio questi aspetti attraverso alcuni valori riassuntivi.

2.6.1 Mean Amplitude of Glycemic Excursions (MAGE)

Forse uno dei più famosi ed accreditati indici di variabilità glicemica, il MAGE [22] viene calcolato mediando il valore di tutte le escursioni glicemiche significative, ossia quelle che sono abbastanza ampie rispetto alla metrica proposta contestualmente alla valutazione dell'indice.
Definendo, quindi, $\Delta G(k^*)$ ciascuna delle N_{ex} escursioni individuate si può, facilmente, valutare il MAGE come

$$\mathrm{MAGE} \coloneqq \frac{1}{N_{ex}} \sum_{k^*=1}^{N_{ex}} \Delta G(k^*) \tag{2.14}$$

Tuttavia, nella pratica, questa definizione viene raramente applicata così com'è, preferendo, specie per le applicazioni CGM, una procedura decisamente più complicata, ma più adatta al caso del monitoraggio in continua.
Prima di procedere, si fa presente che le due tecniche di calcolo non sono equivalenti e che, anzi, la seconda permette di estrarre gli ulteriori indici MAGE+ e MAGE-, rispettivamente limitati alle escursioni positive e negative.

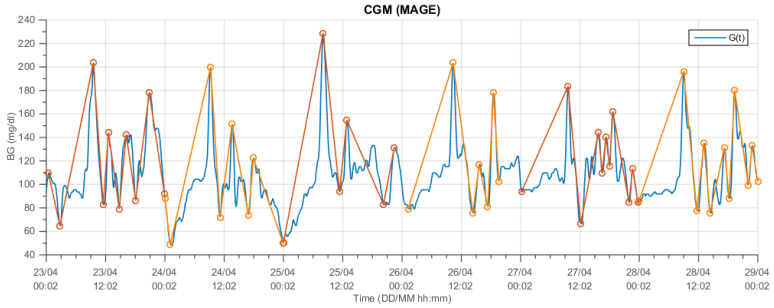


Figura 2.5: Calcolo dell'indice MAGE. Le linee spezzate mostrano la corrispondenza tra il tracciato originale e ciascuna sequenza di escursioni giornaliere.

2.6.1.1 Protocollo operativo per l'estrazione del MAGE

In letteratura si può ritrovare una tecnica per il calcolo del MAGE specifica per i segnali CGM che si può considerare l'attuale gold standard atto a questo scopo [23]. Purtroppo, però, l'algoritmo descritto lascia a desiderare dal punto di vista dell'individuazione delle escursioni significative e, in particolare, dei loro estremi. Sebbene, infatti, la procedura generale con cui questi debbano essere scelti sia perfettamente accettabile, nell'articolo si fa anche presente come essa possa risultare approssimativa e si invita, in calce, all'ispezione visiva dei risultati ottenuti. Poiché si ritiene che la necessità di intervento umano nel contesto di una procedura che ci si aspetterebbe essere automatica sia del tutto inaccettabile, nonostante i risultati ottenuti possano dirsi comparabili, si propone qui una revisione dell'algoritmo mirata a risolvere il problema.
Senza ulteriore indugio, si delineano, dunque, estensivamente i passi necessari per passare dal segnale CGM campionato $G(k)$ al valore finale dell'indice MAGE. Una rappresentazione visiva delle escursioni individuate si può trovare in figura 2.5.
Step 1. Suddividere il segnale $G(k)$ in porzioni $G_d(k)$, ciascuna corrispondente al d-esimo degli N_{days} giorni che costituiscono la durata totale dell'acquisizione. Nel seguito si asumme che $G_d(k)$, di lunghezza N_d, sia scandito a partire da $k = 1$ fino a $k = N_d$.
Step 2. Per $d = 1, \ldots, N_{days}$ individuare l'insieme finale $\mathcal{J}_d$ dei *turning point*, cioè degli estremi delle escursioni significative, secondo la procedura seguente.

1. Calcolare la deviazione standard giornaliera σ_d di $G_d(k)$ con la consueta formula 2.2.

2. Trovare tutti gli estremi locali presenti nella d-esima porzione di $G(k)$, ricordando che un punto $G_d(k)$ si definisce estremo locale (rispettivamente, massimo o minimo) se

$$\begin{cases} G_d(k) > G_d(k+1) \\ G_d(k) > G_d(k-1) \end{cases} \quad \vee \quad \begin{cases} G_d(k) < G_d(k+1) \\ G_d(k) < G_d(k-1) \end{cases}$$

3. Inserire tutti gli estremi locali nell'insieme dei turning point $\mathcal{J}_d$. Si assume che $\mathcal{J}_d$ contenga, passo dopo passo, un numero variabile di elementi $G_d(k^*)$ per $k^* = 1, \ldots, N_{turn}$

4. Poiché la definizione di estremo locale richiede l'esistenza contemporanea di entrambi i vicini, ma non è detto che il primo e l'ultimo campione di $G_d(k)$ non siano, a loro volta, parte di escursioni significative, essi vengono aggiunti post hoc a $\mathcal{J}_d$.

5. Rimuovere da $\mathcal{J}_d$ i punti che non sono significativamente diversi, contemporaneamente, tra entrambi i loro vicini. Due turning point adiacenti sono significativamente diversi se la loro differenza è, in modulo, maggiore di una volta la deviazione standard giornaliera σ_d. Si noti che il primo e l'ultimo punto della sequenza non possono essere rimossi a questo punto della procedura, poiché sprovvisti di uno dei due vicini.
   Si fa presente, inoltre, che questo step non deve essere eseguito in sequenza, ovvero che, prima di rimuovere alcunché, tutto l'insieme $\mathcal{J}_d$ deve essere stato esaminato.

6. Rimuovere gli elementi di $\mathcal{J}_d$ che si trovano nel mezzo di una sequenza monotona (crescente o decrescente) di tre punti e, contemporaneamente, aggiustare la posizione dei turning point sopravvissuti in modo che, effettivamente, siano minimi o massimi locali. A differenza del passo precedente, qui i turning point devono essere esaminati in sequenza, da sinistra verso destra.
   Nello specifico, per ogni $G_d(k^*) \in \mathcal{J}_d$ può accadere che:
   a. $G_d(k^*)$ sia un minimo locale (o vicino a esso), cioè

$$\begin{cases} G_d(k^*) < G_d(k^*+1) \\ G_d(k^*) < G_d(k^*-1) \end{cases}$$

   In questo caso, la posizione dei turning point viene aggiustata effettuando, tassativamente nell'ordine, le seguenti sostituzioni:

$$\begin{aligned} G_d(k^*) &= \min_{G_d(k)\,:\,k^*-1\le k\le k^*+1}\{G_d(k)\} \\ G_d(k^*-1) &= \max_{G_d(k)\,:\,k^*-1\le k\le k^*}\{G_d(k)\} \\ G_d(k^*+1) &= \max_{G_d(k)\,:\,k^*\le k\le k^*+1}\{G_d(k)\} \end{aligned}$$

   b. $G_d(k^*)$ sia un massimo locale (o vicino a esso), cioè

$$\begin{cases} G_d(k^*) > G_d(k^*+1) \\ G_d(k^*) > G_d(k^*-1) \end{cases}$$

   In questo caso, la posizione dei turning point viene aggiustata effettuando, tassativamente nell'ordine, le seguenti sostituzioni:

$$\begin{aligned} G_d(k^*) &= \max_{G_d(k)\,:\,k^*-1\le k\le k^*+1}\{G_d(k)\} \\ G_d(k^*-1) &= \min_{G_d(k)\,:\,k^*-1\le k\le k^*}\{G_d(k)\} \\ G_d(k^*+1) &= \min_{G_d(k)\,:\,k^*\le k\le k^*+1}\{G_d(k)\} \end{aligned}$$

c. $G_d(k^*)$ sia al centro di una sequenza monotona di tre punti, cioè

$$G_d(k^*-1) < G_d(k^*) < G_d(k^*+1)$$
$$\vee$$
$$G_d(k^*-1) > G_d(k^*) > G_d(k^*+1)$$

In questo caso, $G_d(k^*)$ viene semplicemente tolto da $\mathcal{J}_d$.

7. Rimuovere i turning point che non sono significativamente differenti da almeno uno dei vicini.
La procedura deve essere differenziata tra i punti centrali e $G_d(1), G_d(N_{turn})$ che vengono tolti da $\mathcal{J}_d$ se e solo se non sono significativamente diversi, rispettivamente, da $G_d(2)$ e $G_d(N_{turn}-1)$. Questo è sufficiente poiché, a seguito dei passi precedenti, non è possibile si presentino altre casistiche. I punti centrali, invece, vengono esaminati in sequenza, da sinistra verso destra, e rimossi se non sono significativamente diversi da almeno uno dei candidati turning point a loro adiacenti.

8. A questo punto $\mathcal{J}_d$ è l'insieme finale di tutti e soli i turning point per la d-esima giornata.

9. Calcolare il valore delle escursioni glicemiche significative semplicemente sottraendo a ciascun elemento il suo predecessore all'interno di $\mathcal{J}_d$:

$$\Delta G_d(k^*) = G_d(k^*) - G_d(k^*-1),\ k^* = 2, \ldots, N_{turn}$$

Si noti che ciò ha senso perché nell'insieme dei turning point si alternano massimi e minimi locali di $G_d(k)$.

10. Estrarre gli indici giornalieri MAGE+$_d$ e MAGE-$_d$ come segue:

$$\text{MAGE+}_d = \frac{1}{N^+_{turn}} \sum_{\Delta G_d(k*)>0} \Delta G_d(k*)$$

$$\text{MAGE-}_d = \frac{1}{N^-_{turn}} \sum_{\Delta G_d(k*)<0} |\Delta G_d(k*)|$$

dove N^+_{turn} e N^-_{turn} indicano il numero di escursioni rispettivamente positive e negative.

Step 3. Mediare su tutti i valori giornalieri di MAGE+$_d$ e MAGE-$_d$ per ottenere i corrispettivi indici riassuntivi:

$$\text{MAGE+} = \frac{1}{N_{days}} \sum_{d=1}^{N_{days}} MAGE{+}_d$$

$$\text{MAGE-} = \frac{1}{N_{days}} \sum_{d=1}^{N_{days}} MAGE{-}_d$$

Step 4. Calcolare il valore finale dell'indice MAGE:

$$\text{MAGE} = \frac{\text{MAGE+} + \text{MAGE-}}{2}$$

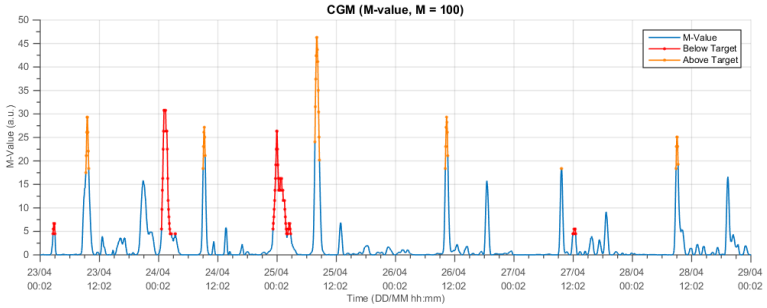


Figura 2.6: Trasformazione M-value. Questo nuovo segnale è colorato in modo da evidenziare la fascia a cui apparteneva ciascun campione prima della trasformazione: azzurro per l'euglicemia, arancio per l'iperglicemia e rosso per l'ipoglicemia

2.6.2 Excursion Frequency (EF)

L'excursion frequency è un indice di più recente applicazione attraverso il quale si forniscono informazioni complementari a quelle portate dall'indice MAGE [24][25][26]. Infatti, laddove il MAGE tiene conto soltanto del modulo (e marginalmente del segno) delle escursioni significative, l'excursion frequency ne considera il numero medio giornaliero, limitatamente a quelle che, in valore assoluto, superano la soglia dei 75 mg/dl:

$$\mathrm{EF} := \sum_{|\Delta G(k^*)|>75} \frac{1}{N_{days}} \tag{2.15}$$

dove $\Delta G(k^*)$ indica la generica escursione significativa rispetto alle definizioni viste per il mage.

2.7 Indici basasi su trasformazioni empiriche della scala glicemica

Gli indici presentati finora hanno la caratteristica comune di poter essere estratti direttamente dal tracciato CGM, senza mai agire sul livello di glucosio acquisito dal sensore. Negli anni sono state elaborate, però, diverse tecniche atte a enfatizzare alcune caratteristiche notevoli del segnale tramite trasformazioni non lineari. L'obiettivo generale sembra essere quello di amplificare, qualora siano presenti, eventuali picchi in uscita dal range euglicemico $[70, 180]$ mg/dl. Di seguito se ne presentano alcuni che, nonostante siano evidentemente frutto di considerazioni del tutto empiriche, hanno avuto vari gradi di successo in letteratura.

2.7.1 M-value

L'M-value si calcola come

$$M_R := \frac{1}{N} \sum_{k=1}^{N} \left[1000 \cdot \left| \log_{10} \left(\frac{G(k)}{R} \right) \right|^3 \right] \tag{2.16}$$

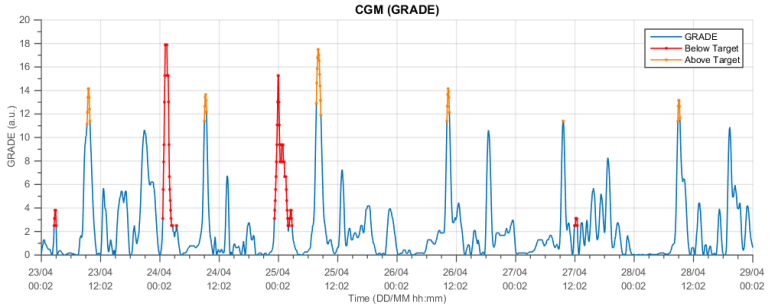


Figura 2.7: Trasformazione GRADE. Questo nuovo segnale è colorato in modo da evidenziare la fascia a cui apparteneva ciascun campione prima della trasformazione: azzurro per l'euglicemia, arancio per l'iperglicemia e rosso per l'ipoglicemia

dove R è un parametro da fissare, ma per cui si suggerisce un valore pari a 100. Tale trasformazione, come si vede in figura 2.6, risulta in un segnale fortemente appiattito in concomitanza dei campioni $G(k) \in [70, 180]$ e notevolmente amplificato all'esterno, specie nel caso dell'iperglicemia.

2.7.2 Glycemic Risk Assessment Diabetes Equation (GRADE)

La trasformazione seguente [27]

$$\mathrm{GRADE}(k) \coloneqq 425 \cdot \left[\log_{10}\left(\log_{10}\left(\frac{G(k)}{18}\right) + 0.16\right)\right]^2 \tag{2.17}$$

sebbene qualitativamente molto simile a quella utilizzata per la definizione dell'M-value, porta, rispetto a quest'ultima, a un minore appiattimento in concomitanza di campioni $G(k)$ riferiti alla fascia euglicemica (vedi figura 2.7).

A partire dall'equazione 2.17, si possono calcolare quattro indici, limitando opportunamente la valutazione di $\mathrm{GRADE}(k)$:

$$\begin{aligned}
\mathrm{GRADE} &\coloneqq \frac{1}{N}\sum_{k=1}^{N}\mathrm{GRADE}(k) \\
\mathrm{GRADE}_{eu} &\coloneqq \frac{\sum_{\Omega_{eu}}\mathrm{GRADE}(k)}{\sum_{\Omega}\mathrm{GRADE}(k)} \\
\mathrm{GRADE}_{hyper} &\coloneqq \frac{\sum_{\Omega_{hyper}}\mathrm{GRADE}(k)}{\sum_{\Omega}\mathrm{GRADE}(k)} \\
\mathrm{GRADE}_{hypo} &\coloneqq \frac{\sum_{\Omega_{hypo}}\mathrm{GRADE}(k)}{\sum_{\Omega}\mathrm{GRADE}(k)}
\end{aligned} \tag{2.18}$$

dove

$$\begin{aligned}
\Omega_{eu} &\coloneqq \{k : 70 \leq G(k) \leq 180\} \\
\Omega_{hyper} &\coloneqq \{k : G(k) > 180\} \\
\Omega_{hypo} &\coloneqq \{k : G(k) < 70\} \\
\Omega &\coloneqq \Omega_{eu} \cup \Omega_{hyper} \cup \Omega_{hypo}
\end{aligned} \tag{2.19}$$

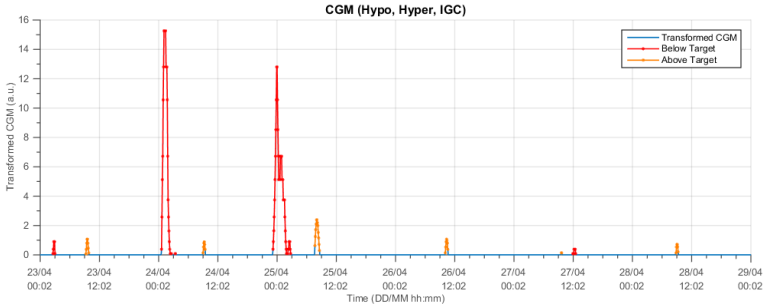


Figura 2.8: Trasformazione per Hypo e Hyper Index. Questo nuovo segnale,ottenuto dalla giustapposizione delle due trasformazioni in 2.20, è colorato in modo da evidenziare la fascia a cui apparteneva ciascun campione prima della trasformazione: azzurro per l'euglicemia, arancio per l'iperglicemia e rosso per l'ipoglicemia.

definiscono gli insiemi degli istanti di campionamento per cui $G(k)$ aopartiene a ciascuna delle tre fasce.

2.7.3 Indici di controllo glicemico

I prossimi indici, finalizzati a caratterizzare la qualità del controllo glicemico, sono estratti a partire da una trasformazione che, in un certo senso, si può intendere come l'approssimazione di un algoritmo di *peak detection*: la fascia euglicemica è spinta fortemente verso lo zero, mentre sono amplificati picchi in uscita da essa (vedi figura 2.8).

Le formule per il calcolo, in cui sono stati direttamente inseriti i parametri di default e in cui si considerano $70, 180$ come estremi del range euglicemico, sono le seguenti:

$$\begin{aligned} \text{Hypo Index} &:= \frac{1}{30 \cdot N} \sum_{\Omega_{hypo}} [70 - G(k)]^2 \\ \text{Hyper Index} &:= \frac{1}{30 \cdot N} \sum_{\Omega_{hyper}} [G(k) - 180]^{1.1} \end{aligned} \tag{2.20}$$

con Ω_{hypo} e Ω_{hyper} definiti come in (2.19).
La somma dei due indici, infine, costituisce il cosiddetto Index of Glycemic Control:

$$\text{IGC} := \text{Hypo Index} + \text{Hyper Index} \tag{2.21}$$

2.8 Indici di rischio statico

Concettualmente molto simili a quelli presentati nella sezione precedente, anche gli indici di rischio statico [28] vengono calcolati a valle di una trasformazione empirica della scala glicemica. Tuttavia, vale la pena prenderli in considerazione separatamente perché affrontano esplicitamente il problema dell'asimmetria rispetto alla media dell'andamento del glucosio. In particolare, assumendo per $G(t)$ il codominio $[20, 600]$ mg/dl, è evidente come la fascia euglicemica $[70, 180]$ mg/dl non solo non ne occupi la regione di mezzo, ma non abbia neppure un

centroide simile (circa 300 mg/dl contro 100 mg/dl). Inoltre, sebbene la condizione di ipoglicemia ($G(t) < 70$ mg/dl) sia più grave, specie nel breve termine, di quella di iperglicemia ($G(t) > 180$ mg/dl) a quest'ultima è assegnata una porzione ben maggiore del codominio stesso, sia per estensione, sia per valore assoluto.
Dunque, con l'obiettivo di, in un certo senso, normalizzare la scala glicemica è stata messa a punto la seguente trasformazione (si omette la dipendenza dal tempo di $G(t)$):

$$f(G) \coloneqq 1.509 \cdot \left[(\log G)^{1.084} - 5.381\right] \tag{2.22}$$

La funzione $f(G)$, valutata per valori significativi di G, esibisce alcune ottime proprietà:

1. mappa il range $[20, 600]$ mg/dl in [-3.16, +3.16];
2. vale 0 in concomitanza del valore $G = 112.5$;
3. vale +0.88 sulla soglia di iperglicemia;
4. vale −0.88 sulla soglia di ipoglicemia;

In altre parole, tramite $f(G)$ si trasforma la scala glicemica in modo che essa sia circa simmetrica (per estensione e per valori assunti, ma non necessariamente in distribuzione) rispetto a quello che si può considerare il suo zero clinico, cioè un valore consistente con una tranquilla permanenza in regione di euglicemia.
Per amplificare, poi, uscite significative dalla fascia $[70, 180]$, si definisce l'ulteriore trasformazione

$$r(G) \coloneqq 10 \cdot [f(G)]^2 \tag{2.23}$$

La funzione $r(G)$ indica, dunque, il rischio statico e, come si nota dalla sua formulazione, assume valori particolarmente elevati nel caso in cui $|f(G)| > 1$, cioè per $G \geq 192$ oppure $G \leq 65.6$, cioè in concomitanza di uscite nette dalla fascia euglicemica.
Data la diversa gravità e le differenti caratteristiche delle condizioni di ipoglicemia e iperglicemia, $r(G)$ viene ridefinita separando i due casi

$$r(G) \coloneqq \begin{cases} r_l(G) & \text{, se } G \leq 112.5 \\ r_h(G) & \text{, se } G > 112.5 \end{cases} \tag{2.24}$$

dove, ovviamente,

$$r_l(G) \coloneqq \begin{cases} r(G) & \text{, se } G \leq 112.5 \\ 0 & \text{, altrimenti} \end{cases} \tag{2.25}$$

$$r_h(G) \coloneqq \begin{cases} r(G) & \text{, se } G > 112.5 \\ 0 & \text{, altrimenti} \end{cases} \tag{2.26}$$

L'effetto del passaggio a $r(G)$ si può apprezza in figura 2.9.
Giunti a questo punto si possono calcolare i due indici Low Blood Glucose Index e High Blood Glucose Index semplicemente mediando $r_l(G)$ e $r_h(G)$ su tutti i campioni trasformati a disposizione (compresi quelli nulli)

$$\text{LBGI} \coloneqq \frac{1}{N} \sum_{k=1}^{N} r_l(G(k)) \tag{2.27}$$

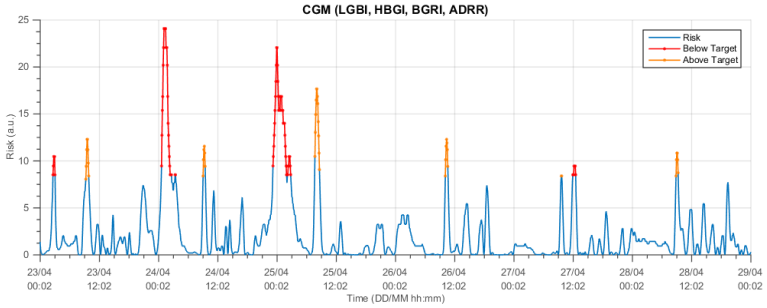


Figura 2.9: Rischio statico. Questo nuovo segnale è colorato in modo da evidenziare la fascia a cui apparteneva ciascun campione prima della trasformazione: azzurro per l'euglicemia, arancio per l'iperglicemia e rosso per l'ipoglicemia.

$$\mathrm{HBGI} := \frac{1}{N}\sum_{k=1}^{N} r_h(G(k)) \tag{2.28}$$

Inoltre, la loro somma costituisce il Blood Glucose Risk Index

$$\mathrm{BGRI} := \mathrm{LBGI} + \mathrm{HBGI} \tag{2.29}$$

Infine, sempre a partire da $r_l(G)$ e $r_h(G)$, si può calcolare un ultimo indice, più complicato, noto come Average Daily Risk Range

$$\mathrm{ADRR} := \frac{1}{N_{days}}\sum_{j=1}^{N_{days}}\left[\max_k[r_l^j(G(k))] + \max_k[r_h^j(G(k))]\right] \tag{2.30}$$

dove la notazione $\max_k[r_l^j(G(k))]$ indica il massimo della funzione r_l limitata alla sola j-esima giornata di osservazioni.

2.9 Indici basati sui momenti dell'immagine

Una categoria di indici abbastanza utilizzata in letteratura [24][25][26] è quella che si basa sulle proprietà dell'immagine binaria ottenuta ponendo a 1 i pixel compresi tra la curva del tracciato CGM e la riga orizzontale corrispondente al minimo livello di glucosio registrato (vedi figura 2.10).
Di fronte all'evidente arbitrarietà, specie sulla misura dei pixel, con cui si può effettuare la conversione segnale-immagine, si è scelto di ricorrere a una collezione di sette indici indipendenti, i *Moment Invariant* di Hu [29], che sono il più possibile invarianti rispetto alla traslazione, rotazione e scalamento. Si tratta di combinazioni non lineari dei momenti centrali normalizzati di ordine 2 e 3, facilmente calcolabili a partire dalla generica definizione del momento centrale $(p+q)$-esimo di un'immagine binaria con intensità nel punto (x,y) pari a $f(x,y)$:

$$\mu_{pq} := \sum_x \sum_y [(x-x_g)^p + (y-y_g)^q] f(x,y) \tag{2.31}$$

dove (x_g, y_g) sono le coordinate del baricentro dell'immagine.
Utilizzando l'equazione 2.31, la versione normalizzata di μ_{pq} si calcola come

$$\eta_{pq} := \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{2.32}$$

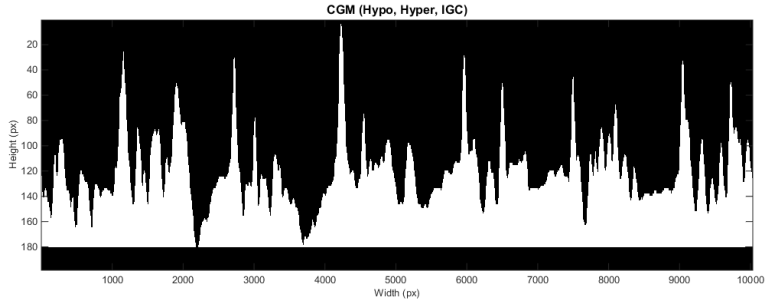


Figura 2.10: Segnale CGM come immagine binaria. I pixel bianchi corrispondono al valore 1, quelli neri al valore 0.

con $\gamma := \dfrac{p+q}{2} + 1$.
A questo punto, i Moment Invariant di Hu sono:

$$\begin{aligned}
\phi_1 &:= \eta_{20} - \eta_{02} \\
\phi_2 &:= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
\phi_3 &:= (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2 \\
\phi_4 &:= (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2 \\
\phi_5 &:= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) + \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
\phi_6 &:= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + \\
&\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi_7 &:= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) + \\
&\quad + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)
\end{aligned} \tag{2.33}$$

Si noti che l'ultimo indice gode di una proprietà in più rispetto agli altri: rimane costante anche specchiando l'immagine.

2.10 Altri parametri clinici del paziente

Sebbene non si tratti, tecnicamente, di indici estratti da dati di sensore CGM, si include, in coda a questo capitolo, un breve riferimento ad alcune variabili di interesse circa lo stato clinico del paziente.
Rientrano nella categoria dei parametri clinici di base alcune informazioni sul paziente che, nonostante possano essere ottenute in maniera agevole, sono molto significative per costruirne una sorta ritratto. Si tratta di:

- età;
- sesso;
- indice di massa corporea;
- circonferenza della vita.

Queste quattro quantità sono sufficienti, a fronte di alcune considerazioni di natura fisiologica e fisionomica, a caratterizzare la condizione fisica del paziente limitatamente a quanto si possa desumere dal suo aspetto esteriore. Un esempio banale in proposito, è ci si aspetta che, a parità di indice di massa corporea, la circonferenza della vita di un uomo sia maggiore di quella di una donna.
Chiaramente, c'è un forte limite a quanto si può apprendere da queste semplici variabili, ma, dato il costo quasi nullo (in tutti i sensi) della loro acquisizione, può essere conveniente includerle per aumentare le informazioni a disposizione in fase di analisi.
Altri parametri clinici devono essere appositamente acquisiti tramite operazioni, in genere, non a costo zero. Si portano ad esempio quelli che verrano utilizzati in fasi successive di questo lavoro:

- la pressione sanguigna (sistolica e diastolica) richiede di avere a disposizione uno sfigmomanometro e qualcuno che se ne sappia servire efficacemente;
- la frequenza del battito cardiaco necessita, anch'essa, di un apposito strumento per la misurazione;
- il glucosio a digiugno (*fasting glucose*) può essere acquisito, tramite prelievo e analisi del sangue, solo dopo un periodo prolungato di digiuno da parte del soggetto.

Come si può notare, in tutti i casi c'è una componente di costo per il paziente, l'operatore o la struttura sanitaria in termini di tempo, equipaggiamento o disagio.

2.11 Uso clinico degli indici di GV

L'applicazione in ambito clinico degli indici di GV è ancora oggetto di investigazione. Infatti, come si è detto, nessuno degli indici, pur essendo stato caratterizzato dal punto di vista sperimentale in lavori come quelli citati nel corso di questo capitolo, è emerso come gold standard. Per far fronte a questa difficoltà, l'attenzione si è concentrata sulla ricerca di un metodo per combinare le informazioni derivanti da diversi indici di GV.
La strada più promettente in questo senso sembra essere quella di inserire il problema in un'ottica di classificazione, sfruttando gli indici stessi come feature. La tendenza prevalente, inoltre, sembra essere quella di includere quante più variabili possibili e, solo in un secondo momento, operare una selezione automatica per ridurne il numero. Questo tipo di approccio, tuttavia, spesso degenera in un'inclusione sregolata di metriche, alcune delle quali chiaramente destinate al fallimento. È questo il caso di lavori come [24] e [30], dove, accanto a indici ragionevoli e accreditati come il MAGE compare, per esempio, l'ampiezza dei primi coefficienti della Discrete Fourier Transform che, come è noto, non sono indicativi per segnali non stazionari [31]. Un'altra caratteristica comune a indagini di questo tipo è quella di adottare un approccio poco rigoroso alla scelta dei classificatori: da [24], [30], [25] e [32] non emerge alcun filo conduttore che giustifichi, anche approssimativamente, perché si è pensato di ricorrere a una tecnica piuttosto che ad un'altra. Dai risultati ottenuti, però, si può dedurre che le Support Vector Machine siano piuttosto adatte al problema in esame, in quanto spesso applicate con successo.

Data la natura della questione che richiede, da un lato, l'applicazione di metodologie piuttosto avanzate dal punto di vista matematico e, dall'altro, il raffronto con la pratica clinica, spesso vengono coinvolti anche medici nella messa a punto dei protocolli sperimentali [26], [33]. Tuttavia, se è vero che non si può prescindere dall'esperienza maturata dagli esperti del settore nel trattamento e caratterizzazione del diabete, il loro contributo deve essere mediato dall'attento scrutinio dell'ingegnere che, più avvezzo a questioni di carattere teorico, deve assicurarsi di non falsare i risultati, introducendo un bias nei processi matematici di sua competenza. Per esempio in [33] si può vedere come sia la messa a punto delle regole di decisione sia la divisione in classi dei soggetti siano state effettuate dallo stesso esperto diabetologo. Si tratta di un *faux pas* subdolo e di difficile individuazione, ma dalle enormi conseguenze sulla affidabilità dei risultati ottenuti. Infatti, in questo contesto si può dire che le prestazioni non siano riferite tanto all'attitudine del classificatore nel riconoscere i pazienti che presentano un certo stato metabolico, ma alla fedeltà con cui esso riproduce la forma mentis del particolare clinico. Sempre a questo proposito, si nota anche come l'attenzione venga spesso sposta sulla definizione di livelli di controllo glicemico, procedimento, peraltro, abbastanza soggettivo (vedi anche [24]).
Per quanto riguarda, poi, la selezione della combinazione di indici più indicata per la descrizione dello stato metabolico di un soggetto, in letteratura sono proposti gli approcci più disparati. Uno particolarmente interessante, almeno dal punto di vista dell'originalità, è quello presentato in [18], basato sulla rappresentazione sparsa delle componenti principali. Purtroppo, sono dubbie le implicazioni dell'utilizzo di questa tecnica per la classificazione dei pazienti affetti da diabete di tipo 2. Infatti, l'analisi ulteriore proposta in [32] sembra mostrare soltanto che l'approssimazione introdotta tramite SPCA (*Sparse Principal Component Analysis*) non influisce sulle prestazioni del classificatore il che, però, è abbastanza scontato, vista la relazione tra SPCA e PCA semplice.
Si cita, infine, [34] per mettere in luce l'esistenza di un approccio alternativo alla feature selection per l'individuazione degli indici significativi. In questo lavoro, infatti, si preferisce combinare linearmente alcune metriche ritenute (per la verità arbitrariamente) significative in un unico valore, in maniera concettualmente simile al J-index [19].

2.12 Scopo della tesi e presentazione dei contenuti

Alla luce di quanto detto nei capitoli 1 e 2, emerge chiaramente come la caratterizzazione dello stato metabolico di un paziente tramite indici di GV rimanga un problema aperto. Infatti, non è ben chiaro né quali metriche (tra le svariate decine che sono state proposte) siano le più appropriate, né come esse debbano essere utilizzate. Inoltre, non è ancora stato dimostrato quali siano, nello specifico, i problemi per i quali sia utile ricorrere alla GV in sostituzione o in supporto delle tecniche diagnostiche convenzionali. Addirittura, molti tentativi in tal senso risultano fondati sull'euristica e sono, a volte, viziati da inesattezze metodologiche.
Con questa tesi ci si propone, dunque, di mettere a punto un sistema diagnostico efficace basato sull'estrazione degli indici di GV dal tracciato CGM in

maniera non invasiva. Nella fattispecie, opportune tecniche di classificazione saranno rigorosamente applicate a un problema finora mai affrontato: la distinzione tra IGT e T2D. Tale scopo sarà perseguito combinando opportunamente le informazioni provenienti dal sensore CGM e alcuni parametri clinici di base del paziente, in modo tale da minimizzare o, addirittura, annullare l'invasività della procedura che, pure, permetterà di diagnosticare lo stato di avanzamento del diabete di tipo 2 con risoluzione molto fine (pochi giorni, rispetto ai mesi richiesti da tecniche basate sull'HbA1c).
Dopo questi due capitoli introduttivi, dunque, si presenteranno formalmente gli aspetti salienti delle tecniche di classificazione utilizzate (capitolo 3) e le metodologie più significative per training, test, validazione e feature selection (capitolo 4). Nel capitolo 5 si passerà, poi, alla descrizione e discussione di tutti gli aspetti preliminari all'applicazione diretta dei classificatori, le cui performance saranno riportate e analizzate nel capitolo 6. L'ultimo capitolo sarà dedicato, come di consueto, alle conclusioni e alla delineazione di alcuni possibili sviluppi futuri.

Capitolo 3

Classificazione: tecniche principali

3.1 Aspetti introduttivi: classificazione e regressione

I problemi di classificazione appartengono, accanto a quelli di regressione, alla grande categoria dell'apprendimento supervisionato (*supervised learning*). Si tratta di una branca del machine learning caratterizzata dalla conoscenza della corrispondenza input-output tra un insieme di dati in ingresso x appartenenti al *feature space* $\mathcal{X}$ e le relative uscite y, appartenenti, invece, al *target space* $\mathcal{Y}$. In termini formali, l'obiettivo del supervised learning è utilizzare le coppie ordinate $(x, y) \in \mathcal{X} \times \mathcal{Y}$ a disposizione per stimare una funzione $\hat{f}(x^*) = \hat{y}$ in modo tale che $\hat{y} \simeq y^*$, dove y^* è il reale corrispondente dell'ingresso x^* nel target space. La natura degli elementi di $\mathcal{Y}$ è il fattore chiave nella distinzione tra classificazione e regressione.
Un problema si dice di classificazione se $y \in \{C_1, C_2, \ldots, C_n\}$, ovvero se y può assumere un valore all'interno di un set discreto e finito di classi (qui denotate dal simbolo C_i per $i = 1, \ldots, n$). In questo caso y si configura come variabile qualitativa, nel senso che l'etichetta (*label*) C ha soltanto il compito di identificare la categoria a cui y appartiene. L'importante, cioè, non è il suo valore, ma il fatto che coppie (x, y) appartenenti alla stessa classe abbiano la stessa etichetta. Questo non deve, però, far pensare che la scelta dell'etichetta sia indifferente o casuale. Infatti, per le label, esistono formati appropriati che dipendono, di volta in volta, dalla situazione in esame e dal metodo risolutivo. Nel seguito, quando necessario, si farà sempre riferimento a quale convenzione sia meglio adoperare per ciascuna tecnica presentata.
Diversamente, se y può assumere valori lungo tutto l'asse reale, si parla di regressione. In questo caso, y si definisce variabile quantitativa poiché rappresenta una caratteristica dei dati per cui è possibile e desiderabile definire un ordinamento in $\mathbb{R}$.
La versione estesa di questo breve excursus introduttivo si può trovare in [35], a cui si rimanda anche per approfondimenti circa i metodi di classificazione presentati nelle prossime sezioni.

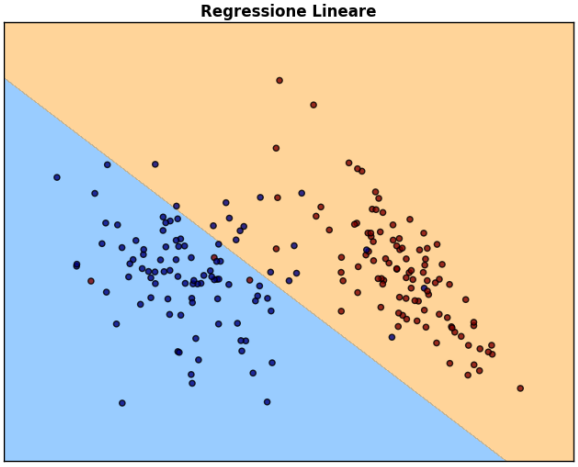


Figura 3.1: Classificatore lineare basato sulla regressione. Le regioni di decisione sono stimate a partire dai dati rappresentati. Il colore è indicativo della classe di appartenenza.

3.2 Classificazione lineare

La maniera più semplice in cui si possa pensare di suddividere in classi un certo insieme di dati $x \in \mathcal{X} \subset \mathbb{R}^m$ è quello di separare lo spazio in regioni tramite iperpiani. Ciascun iperpiano divide lo spazio in due semispazi, a ciascuno dei quali si può assegnare un segno, a seconda che sia rivolto nel verso in cui punta il vettore normale all'iperpiano oppure in verso opposto. La classe da assegnare a un punto x si decide, quindi, sulla base della regione occupata da un punto x, ovvero del segno del suo prodotto scalare con il vettore normale a ciascun iperpiano.
Prima di concentrarsi sulla formalizzazione del problema, si pensi al caso semplice in cui $m = 2$, cioè in cui i dati $x \in \mathcal{X}$ sono rappresentabili ponendo, rispettivamente, in ascissa e ordinata i valori della loro prima e seconda componente. Si immagini, inoltre, di dover considerare solo due classi.
Come è noto, nel caso di $\mathbb{R}^2$ gli iperpiani sono rette e, per una basilare definizione della geometria euclidea, ciascuna retta divide il piano in due semipiani. Una volta scelta l'orientazione del versore che definisce la direzione perpendicolare alla retta stessa, si possono considerare tutti i punti che si trovano nel semipiano puntato dal versore come appartenenti a una classe e gli altri come appartenenti all'altra. Il problema si sposta, quindi, sul trovare la posizione di tale retta nel piano per ottenere un risultato simile a quello di figura 3.1.

Dato che il caso multiclasse esula dai fini di questo lavoro, si presenta solo la formulazione matematica relativa a due sole classi. L'estensione non è particolarmente difficile, ma appesantisce di molto la notazione e richiede alcune accortezze in più.
Sia $X \in \mathbb{R}^{n \times m}$ la matrice che contiene, per righe, gli elementi $x_i \in \mathcal{X} \subset \mathbb{R}^m, i = 1, \ldots, n$ e sia Y il vettore che contiene, nell'ordine opportuno, le label $y_i \in \{C_1, C_2\}, i = 1, \ldots, n$. Per stimare l'iperpiano si effettua una procedura in due passi: prima una regressione lineare di $\mathcal{X}$ su $\mathcal{Y}$ come se le label y variassero

su tutto $\mathbb{R}$ e, poi, una sogliatura (arbitraria o attraverso una funzione costo) dei valori così ottenuti sulla base della quale distinguere tra le classi.
Per l'iniziale regressione, il modello parametrico di interesse è, dunque,

$$y = w^{\mathsf{T}} x + b \tag{3.1}$$

dove $w \in \mathbb{R}^m$ è il vettore dei parametri (in gergo, *weight vector*) e $b \in \mathbb{R}$ rappresenta il bias.
Per snellire la notazione, in genere, si effettua la seguente banale trasformazione:

$$\begin{aligned} y &= w^{\mathsf{T}} x + b = \\ &= \begin{pmatrix} w_1 \dots w_m \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} + b = \\ &= \begin{pmatrix} b\ w_1 \dots w_m \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{pmatrix} = \\ &= w'^{\mathsf{T}} x' \end{aligned} \tag{3.2}$$

Con abuso di notazione, nel seguito spesso si tralascerà di indicare l'apice $'$, confidando che il contesto sia sufficiente a risolvere le ambiguità.
Combinando i dati a disposizione e ponendoli nel formato dell'equazione 3.1, si ottiene il modello

$$\begin{gathered} Y = X' w' \\ \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1\ x_1^{\mathsf{T}} \\ \vdots \\ 1\ x_n^{\mathsf{T}} \end{pmatrix} \begin{pmatrix} 1 \\ w_1 \\ \vdots \\ w_m \end{pmatrix} \end{gathered} \tag{3.3}$$

A questo punto, w' viene stimato tramite minimi quadrati lineari non pesati:

$$\begin{aligned} \hat{w'} &= \underset{w'}{\operatorname{argmin}} \left\{ \left\| Y - X' w' \right\|^2 \right\} = \\ &= (X'^{\mathsf{T}} X)^{-1} X'^{\mathsf{T}} Y \end{aligned} \tag{3.4}$$

La predizione del modello $\hat{Y}$ si ottiene combinando le equazioni 3.3 e 3.4:

$$\hat{Y} = X' \hat{w'} = X' (X'^{\mathsf{T}} X)^{-1} X'^{\mathsf{T}} Y \tag{3.5}$$

A differenza degli elementi di Y, per cui gli unici valori possibili si ricordano essere C_1 e C_2, $\hat{Y}$ è un vettore a valori reali. Si rende necessaria, dunque, una sogliatura tale per cui

$$\begin{cases} \hat{y_i} = C_1 & , \text{ se } \hat{y_i} \gtrless t_h \\ \hat{y_i} = C_2 & , \text{ se } \hat{y_i} \lessgtr t_h \end{cases} \tag{3.6}$$

dove $t_h \in \mathbb{R}$ rappresenta il valore di soglia che si potrebbe scegliere come la media aritmetica tra C_1 e C_2 o, meno banalmente, grazie a una qualche funzione

costo.
Una maniera alternativa per risolvere il problema consiste in una scelta opportuna della codifica per le label C_i. Come si accennava nell'introduzione a questo capitolo, infatti, esiste una certa libertà nella definizione dell'effettivo valore delle etichette che, quindi, possono essere adattate per rispondere, di volta in volta, alle esigenze particolari di ciascun metodo. In questo caso, l'esigenza fondamentale è evitare di trovare esplicitamente t_h.
Per far questo, innanzi tutto, si noti come il processo risolutivo finora descritto non sfrutti in nessun modo il fatto che gli elementi y_i e w_i siano scalari. Dunque, è ammesso modificare le label in modo tale che ciascuna classe sia designata dalla posizione dell'unico 1 in un vettore di zeri. Tale tecnica prende il nome di *indicator matrix coding*.
Formalmente, se C_i è l'i-esima di K classi, al valore della sua etichetta (qualunque esso sia) si sostituisce l'i-esimo versore della base canonica di $\mathbb{R}^K$ e un generico elemento di $\mathcal{Y}$ viene trasformato in questo modo:

$$y = C_i \rightarrow y = \underbrace{(0 \dots 0 \overset{\overset{i}{\smile}}{1} 0 \dots 0)}_{K} \tag{3.7}$$

Così facendo, le equazioni 3.3 3.4 e 3.5 rimangono valide, ricordando, però, il cambiamento di dimensioni di Y e di w che ora sono, rispettivamente, una matrice $n \times K$ e una matrice $m \times K$.
L'assegnazione tramite sogliatura dell'equazione 3.6, infine, viene sostituita da un'operazione di massimo: la classe stimata è quella corrispondente alla massima componente del vettore $\hat{y}$ stimato. In formule, con un leggero abuso di notazione e generalizzando al caso di K classi, si può scrivere:

$$\hat{y} = C_i, \text{ se } i = \underset{j}{\operatorname{argmax}} \left(\hat{y}_{(1)} \dots \hat{y}_{(j)} \dots \hat{y}_{(K)}\right) \tag{3.8}$$

dove $\hat{y}_{(j)}$ designa la j-esima componente di $\hat{y}$.

3.3 Fisher's Linear Discriminant

Naturalmente, la tecnica presentata nella sezione precedente non è l'unica per identificare delle superfici di separazioni lineari all'interno di una nuvola di dati. Un'alternativa, concettualmente complementare allo sfruttamento della regressione come step intermedio prima della classificazione, è offerta dall'analisi lineare di Fisher e, in particolare, dal *Fisher's Linear Discriminant* (FLD). Si tratta di una tecnica che ha come punto di partenza alcune assunzione di gaussianità e indipendenza delle classi, ma che si presta anche ad altre interpretazioni. Una di queste è di particolare interesse perché molto ragionevole ed intuitiva, oltre che in grado di fornire una funzione costo del tutto equivalente a quella ottenuta per vie più formali.
Si pensi, dunque, al problema di distinzione fra due classi, immaginando di trovarsi in una situazione favorevole in cui queste siano linearmente separabili (v. figura 3.2).

Ragionevolmente, una buona separazione è caratterizzata dal fatto che i punti afferenti a una classe siano il più possibile distanti da quelli dell'altra e, nel

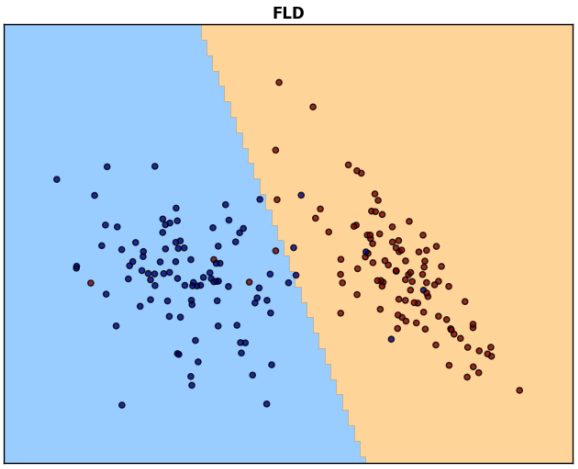


Figura 3.2: Fisher's Linear Discriminant. Le regioni di decisione sono stimate a partire dai dati rappresentati. Il colore è indicativo della classe di appartenenza.

contempo, diano luogo a una sorta di cluster relativamente compatto.
Conseguentemente, si deve cercare un iperpiano tale per cui:

1. i centrodi dei cluster di punti afferenti a ciascuna classe siano il più distante possibile;
2. la varianza all'interno di ciascuna classe sia minore possibile (ossia i cluster siano compatti).

Si noti che i centroidi e la varianza d'interesse, essendo riferiti al target space $\mathcal{Y}$, possono essere definiti come:

$$\begin{aligned} m_k &:= w^\mathsf{T}\mu_k = w^\mathsf{T}\left(\sum_{x_i \in C_k} x_i\right) \\ \sigma_k^2 &:= w^\mathsf{T}\left(\sum_{x_i \in C_k} (x_i - \mu_k)^2\right) w \end{aligned} \tag{3.9}$$

dove, con un lieve abuso di notazione, C_k per $k = 1, 2$ indica l'insieme degli elementi della classe designata dall'etichetta C_k. A questo punto, per rispondere alle esigenze sopra elencate, la funzione costo proposta da Fisher stesso permette di stimare $\hat{w}$ in questo modo:

$$\begin{aligned} \hat{w} &= \underset{w}{\operatorname{argmax}} \left\{ \frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2} \right\} = \\ &= \underset{w}{\operatorname{argmax}} \left\{ \frac{w^\mathsf{T}\Sigma_B w}{w^\mathsf{T}\Sigma_W w} \right\} \end{aligned} \tag{3.10}$$

La sostituzione, banale alla luce dell'equazione 3.9, permette di individuare due matrici che rappresentano la varianza inter-classe (Σ_B, da *between*) e intra-classe

(Σ_W, da *within*), ma che, a differenza di quanto accade per m_k e σ_k^2, sono riferite al feature space $\mathcal{X}$. Grazie ad esse e ad una lieve rimanipolazione (peraltro non troppo rigorosa) della funzione costo per renderla scalare, si potrebbe generalizzare il FLD al caso multiclasse che, però, non viene presentato.
A questo punto, compreso il ragionamento che sta alla base del Fisher's Linear Discriminant, è bene darne un'interpretazione probabilistica.
Siano, dunque, gli elementi delle due classi estratti in maniera indipendente da due distribuzioni gaussiane con medie diverse. In formule, con ovvio significato dei simboli utilizzati:

$$\begin{aligned} \mu_1 &\neq \mu_2 \\ \Sigma \coloneqq \Sigma_1 &= \Sigma_2 \end{aligned} \tag{3.11}$$

Per assegnare un generico elemento $x \in \mathbb{R}^m$ a una classe, si individua quella per cui la probabilità a posteriori $P(y = C_k|x)$ è massima. Le ragioni per questa scelta si potrebbero derivare rigorosamente alla luce di considerazioni sull'errore di classificazione in un'ottica bayesiana. Tuttavia, per gli scopi di questo lavoro, è sufficiente tener conto del fatto che $P(y = C_k|x)$ si può intendere come la probabilità che la classe corretta sia C_k, noto il valore del dato x.
Sotto le ipotesi di gaussianità e indipendenza cui si faceva cenno in precedenza, si può esprimere in forma esplicita la densità di probabilità di x all'interno della classe C_k:

$$p_k(x) = \frac{1}{\sqrt{(2\pi)^m|\Sigma|}}\, e^{-\frac{1}{2}(x-\mu_k)^\intercal \Sigma^{-1}(x-\mu_k)} \tag{3.12}$$

Si può, poi, applicare la regola di Bayes omettendo, come spesso si fa in questi casi, di esprimere per esteso il denominatore, in quanto indipendente dalla scelta della classe.

$$\begin{aligned} P(y = C_k|x = x^*) &= \frac{p_k(x)\pi_k}{p(x)} \\ &\propto p_k(x)\pi_k \end{aligned} \tag{3.13}$$

dove $\pi_k = P(y = C_k)$ (costante in x) rappresenta la probabilità a priori della classe C_k.
Sostituendo l'equazione 3.12 nell'equazione 3.13 si ottiene

$$P(y = C_k|x = x^*) \propto \frac{1}{\sqrt{(2\pi)^m|\Sigma|}}\, e^{-\frac{1}{2}(x-\mu_k)^\intercal \Sigma^{-1}(x-\mu_k)}\pi_k \tag{3.14}$$

Passando al logaritmo si può definire la funzione discriminante $\delta_k(x)$ come segue:

$$\begin{aligned} \delta_k(x) &= \log\left(\frac{1}{\sqrt{(2\pi)^m|\Sigma|}}\, e^{-\frac{1}{2}(x-\mu_k)^\intercal \Sigma^{-1}(x-\mu_k)}\pi_k\right) = \\ &= -\frac{m}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(x-\mu_k)^\intercal\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \end{aligned} \tag{3.15}$$

Dall'equazione 3.15 si possono, ora, togliere i termini comuni tra le classi e utilizzare $\delta_k(x)$ come vera e propria funzione costo:

$$\begin{aligned} \hat{y} &= \underset{C_k}{\operatorname{argmax}}\{\delta_k(x)\} = \\ &= \underset{C_k}{\operatorname{argmax}}\left\{x^\intercal\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^\intercal\Sigma^{-1}\mu_k + \log(\pi_k)\right\} \end{aligned} \tag{3.16}$$

Si tratta di una funzione lineare in x che definisce un iperpiano separatore in $\mathbb{R}^m$, a sua volta lineare. La verifica è banale e, nel caso a due classi, prevede soltanto di porre $\delta_1(x) = \delta_2(x)$; viene, dunque, tralasciata.
Per completezza, si fa presente che se le classi avessero matrici di varianza-covarianza diverse ($\Sigma_1 \neq \Sigma_2$) verrebbe mantenuta, nel passaggio dalla 3.15 alla 3.16, la dipendenza da Σ, dando origine a superfici di separazione quadratiche.

3.4 Regressione logistica

La regressione logistica, a differenza di quanto suggerirebbe il nome, è una tecnica di classificazione lineare che mira a risolvere il problema degli outliers, ossia la tendenza dei due metodi visti fino ad ora a produrre risultati insoddisfacenti nel caso in cui i dati su cui si effettua la stima dei parametri non si presentino in cluster compatti. Tale fenomeno è, purtroppo, intrinseco alla formulazione del classificatore lineare basato sull'approssimazione ai minimi quadrati e del Fisher's Linear Discriminant poiché dipende, in prima approssimazione, dalla non-saturazione delle funzioni costo utilizzate: dati molto distanti dal centroide della classe vengono pesati allo stesso modo degli altri, compromettendo la qualità della stima.
Dal punto di vista formale, la regressione logistica si definisce a partire da un sistema di $K-1$ equazioni (K è il numero delle classi) che impongono la linearità e la limitatezza in $[0,1]$ della posterior:

$$\begin{cases} \log \dfrac{P(y=C_1|x)}{P(y=C_K|x)} = w_{10} + w_1^\mathsf{T} x \\ \log \dfrac{P(y=C_2|x)}{P(y=C_K|x)} = w_{20} + w_2^\mathsf{T} x \\ \dots \\ \log \dfrac{P(y=C_{K-1}|x)}{P(y=C_K|x)} = w_{(K-1)0} + w_{K-1}^\mathsf{T} x \end{cases} \tag{3.17}$$

Si tratta di $K-1$ equazioni espresse in termini di una trasformazione detta *logit*, cioè il logaritmo del rapporto tra la probabilità di successo e di fallimento. Si può vedere che al denominatore di ciascun rapporto si trova una delle probabilità a posteriori, ma quale essa sia non ha alcuna influenza sul modello (intuitivamente, l'ordine delle classi è irrilevante). La verifica di questa proprietà, così come quella del fatto che le probabilità in effetti sommino a uno, viene tralasciata perché laboriosa e qui poco rilevante, ma si potrebbe eseguire facilmente una volta espresso il sistema 3.17 isolando al primo membro le $P(y = C_i|x)$.

La regressione logistica, come forse si sarà intuito dalla formulazione del modello 3.17, è particolarmente favorevole nel caso in cui le classi in gioco siano soltanto due. Infatti, il tutto si riduce a un'unica equazione lineare, da cui si può ricavare la forma esplicita delle posterior:

$$\begin{cases} P(y=C_1|x) = \dfrac{1}{1+e^{-w^\mathsf{T} x}} = \sigma(w^\mathsf{T} x) \\ P(y=C_2|x) = 1 - P(y=C_1|x) = 1 - \sigma(w^\mathsf{T} x) \end{cases} \tag{3.18}$$

La funzione $\sigma(z)$, detta sigmoide, esibisce delle ottime proprietà:

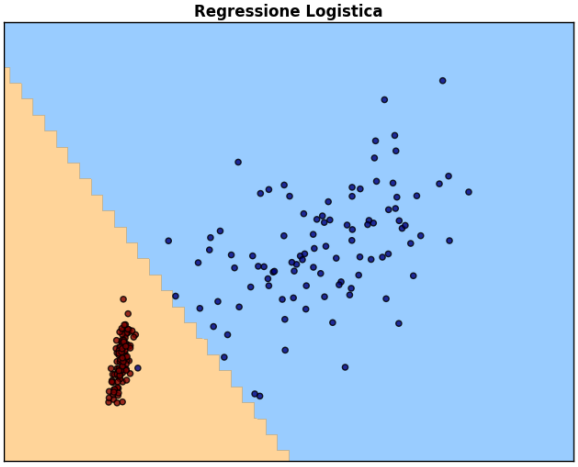


Figura 3.3: Regressione logistica. Le regioni di decisione sono stimate a partire dai dati rappresentati. Il colore è indicativo della classe di appartenenza.

1. ha per codominio $(0, 1)$;
2. $\lim_{z \to -\infty} \sigma(z) = 0$;
3. $\lim_{z \to +\infty} \sigma(z) = 1$;
4. $\dfrac{d\sigma}{dz} = \sigma(z)[1 - \sigma(z)]$.

Per quanto riguarda la stima dei parametri, questa avviene massimizzando la log-verosimiglianza

$$\ell(w) := \sum_{i=1}^{N} \log[P(y = C_k | x = x_i, w)] \tag{3.19}$$

Per semplificare i calcoli, è conveniente assegnare alle label $\{C_1, C_2\}$ i valori $\{1, 0\}$. Così facendo, infatti, si ha che

$$\ell(w) = \sum_{i=1}^{N} \left[y_i \log(\sigma(w^{\mathsf{T}} x)) + (1 - y_i) \log(1 - \sigma(w^{\mathsf{T}} x)) \right] \tag{3.20}$$

Come si può notare, la funzione costo alterna, per ciascun elemento $x_i \in \mathcal{X}$, l'azzeramento di uno dei due addendi. Poiché l'obiettivo è la massimizzazione, una volta stimato $\hat{w}$, il termine non nullo sarà sempre il maggiore dei due, cioè quello corrispondente alla classe a posteriori più probabile per ciascuno punto x_i.
La stima viene effettuata ricercando lo zero della derivata prima della log-verosimiglianza 3.20 tramite il metodo iterativo di Newton-Raphson. Senza scendere nei dettagli, si possono ottenere facilmente delle forme matriciali sia per la derivata prima, sia per la derivata seconda (richiesta, come è noto, dal

metodo di Newton-Raphson in quanto derivata della funzione da azzerare che, a sua volta, è la derivata di $\ell(w)$):

$$\begin{aligned} \frac{\partial \ell}{\partial w} &= X^\mathsf{T}(Y - s) \\ \frac{\partial \ell}{\partial w \partial w^\mathsf{T}} &= -X^\mathsf{T} S X \end{aligned} \tag{3.21}$$

dove s è il vettore dei valori di $\sigma(w^\mathsf{T} x_i)$, mentre S è una matrice diagonale il cui i-esimo elemento non nullo è $\sigma(w^\mathsf{T} x_i)[1 - \sigma(w^\mathsf{T} x_i)]$. In entrambi i casi, all'interno della procedura iterativa, si intende che w è il vettore dei pesi calcolato all'iterazione precedente.
L'aggiornamento di w avviene, infine, in questo modo:

$$\begin{aligned} w_{new} &= w_{old} - \left(\frac{\partial \ell}{\partial w \partial w^\mathsf{T}} \right)^{-1} \frac{\partial \ell}{\partial w} \\ &= w_{old} + (X^\mathsf{T} S X)^{-1} X^\mathsf{T} (Y - s) \end{aligned} \tag{3.22}$$

Un esempio dei risultati ottenibili tramite questa tecnica è presentato in figura 3.3.

3.5 K-Nearest Neighbours

Concettualmente agli antipodi rispetto alle tecniche presentate fino ad ora, il KNN è un metodo completamente non parametrico. Infatti, non si basa sulla stima di un vettore finito di parametri di dimensione indipendente dal numero di dati a disposizione, bensì utilizza proprio la conoscenza di tutti questi ultimi per effettuare la classificazione. In particolare, volendo classificare un nuovo punto $x^* \in \mathbb{R}^m$, data la conoscenza di alcuni dati già assegnati alla classe corretta (x_i, y_i), $i = 1, \ldots, n$, si procede per votazione: si assegna x^* a una classe se il maggior numero di vicini prossimi di x^* afferisce proprio a quella classe.
Una volta fissato il valore di K_{KNN}, cioè il numero di vicini aventi diritto di voto, resta solo da definire una misura della distanza tra punti. In generale, si utilizza la norma p:

$$\|z\|_p = \left[\sum_{j=1}^{N_{comp}} |z_{(j)}|^p \right]^{\frac{1}{p}} \tag{3.23}$$

dove $z_{(j)}$ indica la j-esima delle N_comp componenti del vettore $z \in \mathbb{R}_c^N omp$.
Il KNN permette di ottenere delle superfici di separazione fortemente non lineari, tanto più *smooth* quanto più elevato è il valore di K_{KNN}, la cui forma dipende dal tipo di norma scelto.
Si tratta di un metodo molto potente poiché permettere di risolvere alcuni problemi molto complessi in maniera operativamente molto semplice e con risultati comparabili a tecniche più avanzate. Per un esempio, si veda la figura 3.4.
È bene notare, però, che, per ogni assegnazione di un dato sconosciuto, questo deve essere confrontato con l'intero dataset alla ricerca dei suoi vicini prossimi, il che potrebbe comportare una complessità computazionale notevole. Di converso, questo non accade con i modelli parametrici nei quali, una volta stimati

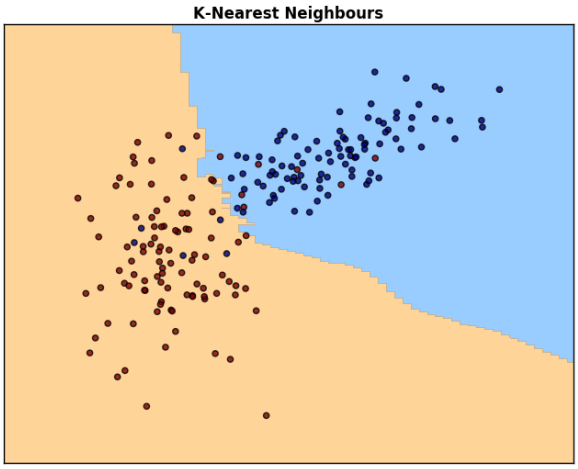


Figura 3.4: K-Nearest Neighbours ($K = 10$). Le regioni di decisione sono stimate a partire dai dati rappresentati. Il colore è indicativo della classe di appartenenza.

(anche onerosamente) i parametri, in genere sono sufficienti poche, semplici operazioni per dar luogo all'assegnazione.
Simili constatazioni si possono fare dal punto di vista dell'occupazione in memoria.

3.6 Metodi di ensemble

I metodi di ensemble sono delle tecniche di classificazione che, per cercare di migliorare le performance, invece di complicare il modello matematico per un unico classificatore, combinano i risultati di molti classificatori deboli (leggi: "semplici").
L'idea è quella delegare l'assegnazione in classi per ciascuna regione del feature space $\mathcal{X}$ ai classificatori che meglio descrivono il comportamento dei dati all'interno della regione stessa. Questo può avvenire, semplicemente, per votazione o, in maniera più robusta, tramite l'introduzione di appropriati pesi indicativi della performance di ciascun classificatore debole. In ogni caso, l'applicazione di un metodo di ensemble preclude quasi sicuramente l'interpretabilità dal punto di vista modellistico dei risultati.
Prima di procedere alla presentazione dei principali metodi di ensemble, è il caso di accennare ad alcuni concetti utili per il seguito.

Stump

Un modo per costruire un classificatore debole è quello di partire da un classificatore parametrico e applicarlo soltanto a un certo subset delle features, cioè delle componenti di un generico dato x.
Sebbene il numero di features di volta in volta utilizzate possa essere scelto a piacere, si predilige, comunemente, uno dei due approcci antipodali seguenti:

1. applicare ripetutamente il classificatore debole a tutte le features (tenendo conto che ogni classificatore sarà riferito a una certa regione del feature space);

2. utilizzare la tecnica dello *stump*, caratterizzata dall'utilizzo del classificatore debole su una feature per volta.

Alberi decisionali

Una struttura molto utilizzata nei metodi di ensemble è quella dell'albero decisionale, che mira a partizionare il feature space $\mathcal{X}$ in un insieme di rettangoli.
Sviscerare completamente la teoria in proposito sarebbe improponibile, vista l'amplissima letteratura a riguardo. Tuttavia, per gli scopi di questo lavoro, è sufficiente comprendere in maniera del tutto generale cosa si intenda per albero decisionale. Scendere ulteriormente nel dettaglio, peraltro, sarebbe inutile perché le diverse implementazioni degli alberi influenzano marcatamente gli algoritmi che su di essi possono essere applicati. Detto questo, a titolo di esempio, si descrive (a parole) il procedimento nel caso del metodo CART per classificazione e regressione tramite strutture ad albero.
Come ci si può aspettare, l'idea è quella di dividere il feature space ricorsivamente a metà tramite una sogliatura (*split*) su una feature alla volta in modo tale che, partendo dalla radice, si possa leggere nelle foglie dell'albero la classe di appartenenza del generico punto x.
Trovare la partizione ottima in senso assoluto sulla base di una qualunque funzione costo, seppur tecnicamente non impossibile, è computazionalmente improponibile. Si ricorre, dunque, a un algoritmo *greedy* per cercare, di volta in volta, lo split che è ottimo per ciascun nodo singolarmente. Fissata una certa funzione costo, per esempio l'errore di classificazione, attraverso cui ottimizzare uno split, si possono calcolare facilmente tutte le coppie (variabile, soglia) che ne rappresentano i minimi locali. Tra tali coppie viene scelta, naturalmente, quella che corrisponde al minimo globale che viene utilizzata, quindi, per aggiungere due nodi figli a quello su cui si è calcolato lo split.
Si continua in questo modo fino al raggiungimento di una certa profondità dell'albero o, più comunemente, fino a quando ogni foglia contiene meno di un certo numero di elementi.
La tendenza, in generale, è quella a costruire alberi molto profondi e sottoporli, poi, al cosiddetto *pruning* (potatura), cioè a una procedura tale per cui si raggiunge un compromesso tra profondità dell'abero e bontà del fit.

3.6.1 Boosting

Il boosting è uno dei metodi di ensemble più potenti. L'idea è quella di aggregare gli output di diversi classificatori deboli, dando maggior credito ai classificatori più precisi sulla porzione di dati a cui sono applicati.
Per rendere immediatamente evidente il funzionamento del boosting, si presenta uno degli algoritmi più popolari, *AdaBoost.M1*, applicato a un problema a due classi con label$C_1, C_2 = -1, +1$.
Siano, come di consueto, $x_i \in \mathcal{X}$ gli elementi del feature space cui corrispondono i valori noti delle etichette $y_i \in \mathcal{Y}$ e sia, inoltre, $g_t(x)$ il t-esimo classificatore

debole utilizzato in sequenza, pesato dal coefficiente α_t. Il classificatore finale, per un generico elemento x, si presenta nella forma:

$$\hat{y} = \text{sign}\left[\sum_{t=1}^{T_{max}} \alpha_t g_t(x)\right] \tag{3.24}$$

dove T_{max} è il numero di classificatori deboli utilizzati.
Il valore di α_t viene calcolato a ogni iterazione dall'algoritmo di boosting in modo da pesare maggiormente l'output di un classificatore preciso, portando, invece, a 0 il contributo di un classificatore completamente inaffidabile.
Si ricorda che un classificatore binario si può considerare completamente inaffidabile se classifica scorrettamente (circa) il 50% dei campioni, cioè se è equivalente al lancio di una moneta equilibrata. Attenzione, però, a non farsi ingannare in un eccesso di zelo: se l'errore è sostanzialmente inferiore al 50% il classificatore ha, in realtà, delle buone performance; basta scambiare le label!
L'algoritmo per ottenere una forma soddisfacente per l'equazione 3.24 procede, dunque, come segue.

1. Inizializzare i pesi $D_{(i)}$, $i = 1, \ldots, n$ relativi a ciascuna osservazione al medesimo valore $\frac{1}{n}$.
   Il vettore D_t, con componenti $D_{(i)}$ permette di controllare quali dati verranno presi maggiormente in condiderazione nel tuning del t-esimo classificatore o, in altri termini, la regione del feature space su esso si deve concentrare.

2. Ripetere le seguenti operazioni per ogni iterazione t per $t = 1, \ldots, T_{max}$.

   (a) Fittare il classificatore debole $g_t(x)$ sui dati, pesando questi ultimi tramite il vettore D_t.
   Dal punto di vista pratico, questo coincide a moltiplicare ciascun addendo della funzione costo per il peso relativo all'elemento in esame.

   (b) Calcolare l'errore di classificazione pesato come

   $$\varepsilon_t = \sum_{i=1}^{n} D_{(i)} \mathbb{I}(y_i \neq g_t(x_i))$$

   dove $\mathbb{I}$ rappresenta la funzione indicatrice che vale 1 se il suo argomento è vero e 0 altrimenti.

   (c) Calcolare il peso α_t del t-esimo classificatore debole all'interno della combinazione finale. In linea di principio è sufficiente che venga calcolato tramite una funzione di ε monotona decrescente in $[0, 0.5]$ che vale 0 se $\varepsilon_t = 0.5$. In pratica, si utilizza la seguente formulazione che, come è facile verificare, risponde al requisito:

   $$\alpha_t = \log\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

   (d) Calcolare le componenti del vettore D_{t+1}, da usare all'iterazione successiva, come

   $$D_{t+1,(i)} = D_{t,(i)} e^{\alpha_t \mathbb{I}(y_i \neq g_t(x_i))}$$

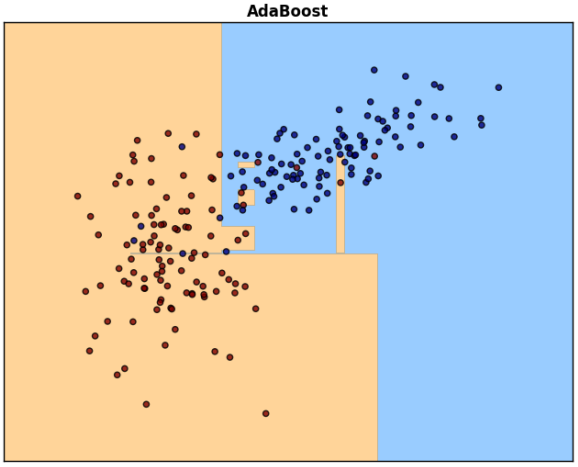


Figura 3.5: AdaBoost. Le regioni di decisione sono stimate a partire dai dati rappresentati. Il colore è indicativo della classe di appartenenza.

Il prossimo classificatore, quindi, si concentrerà maggiormente sulla regione dello spazio in cui i precedenti hanno fallito poiché nella sua funzione costo peseranno molto gli errori sui punti ancora non correttamente classificati.

3. Costruire il classificatore finale come nell'equazione 3.24.

Il risultato dell'algoritmo su un semplice dataset di esempio si può vedere in figura 3.5: si noti la suddivisione evidente in rettangoli.

3.6.2 Bagging

Il bagging è un metodo di ensamble basato sulla costruzione di un classificatore finale come aggregato degli output dello stesso classificatore debole applicato a una collezione di campionamenti bootstrap del dataset a disposizione. Per bootstrap si intende, semplicemente, la costruzione di sottinsiemi (di dimensione costante $n^* < n$) tramite estrazione casuale con reinserimento dei dati x_i, $i = 1, \ldots, n$.
Il classificatore finale, si costruisce, dunque, per votazione da parte dei B diversi classificatori deboli $g_b(x)$:

$$\hat{y} = \text{sign}\left[\sum_{b=1}^{B} \text{sign}(g_b(x))\right] \tag{3.25}$$

dove, per comodità, le classi sono codificate come $\{C_1, C_2\} = \{-1, +1\}$.
A differenza delle tecniche di boosting, che sono messe a punto direttamente sul problema di classificazione, il bagging risulterebbe più adatto per la regressione. Infatti, l'output del regressore finale sarebbe, semplicemente, la media degli output di ciascun regressore debole. Nel caso della classificazione, invece, la tendenza è quella di migliorare le prestazioni di classificatori buoni e peggiorare

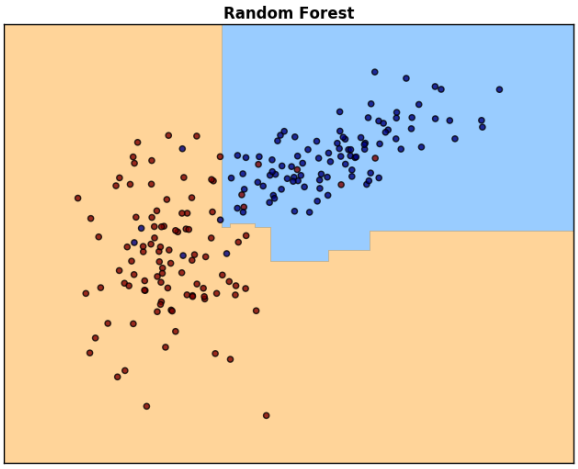


Figura 3.6: Random Forest. Le regioni di decisione sono stimate a partire dai dati rappresentati. Il colore è indicativo della classe di appartenenza.

quelle di classificatori insoddisfacenti.
Questo fenomeno è parzialmente dovuto alle assunzioni di indipendenza tra i classificatori deboli generati tramite bootstrap che, raramente, sono verificate.

3.6.3 Random Forest

La random forest è un metodo di ensemble applicato agli alberi decisionali che ha come scopo quello di migliorare le prestazioni del semplice bagging cercando di risolvere il problema fondamentale da cui quest'ultimo è afflitto: la correlazione tra i classificatori deboli utilizzati.
Ancora una volta, l'idea è quella di mediare molti modelli rumorosi, presumibilmente poco affetti da bias, in modo da ridurne sostanzialmente anche la varianza. Gli alberi decisionali, in tal senso, rappresentano i candidati ideali.
Per considerazioni teoriche in proposito si rimanda alla letteratura. Si preferisce, invece, presentare per esteso un algoritmo di costruzione di una random forest. Così facendo, infatti, si può evidenziare come, all'atto pratico, la scorrelazione tra gli alberi sia ottenuta affiancando al bootstrap sui dati un bootstrap sulle features, estraendole, cioè, in maniera casuale con reinserimento prima di costruire ciascuno dei classificatori deboli.

1. Per ciascuno dei campionamenti bootstrap di dimensione costante n^* estratti dai dati a disposizione, costruire un albero decisionale T_b, per $b = 1, \ldots, B$, ripetendo i passi seguenti per ogni nodo esterno fino al raggiungimento del numero minimo di elementi per foglia.
   (a) Selezionare a caso $m^* < m$ delle m feature.
   (b) Scegliere la miglior coppia (variabile, soglia) tra tutte le possibili.
   (c) Effettuare lo split e accrescere l'albero con due nodi figli.
2. Memorizzare l'ensemble di alberi $\{T_b\}_{b=1,\ldots,B}$.

3. Assegnare un generico dato $x \in \mathcal{X}$, per votazione, alla classe in cui viene inserito dal maggior numero di alberi T_b.

Per avere un'idea dei risultati che si possono ottenere, si veda figura 3.6. Si noti ancora una volta come le regioni di decisione siano date dalla composizione di rettangoli.

3.7 Support Vector Machine

La SVM è una tecnica di classificazione basata sulla separazione lineare tra le classi e costituisce, forse, il miglior metodo parametrico per un problema binario.
L'idea alla base della SVM è quella di dividere il target space $\mathcal{X}$ in due semispazi tramite l'iperpiano separatore che, sotto un'iniziale assunzione di classi perfettamente linearmente separabili, massimizza il margine fra le classi stesse, ovvero la distanza tra l'iperpiano e i punti più vicini ad esso di ciascuna classe.
Si consideri, quindi, di avere a disposizione la consueta conoscenza di alcuni dati $x_i \in \mathcal{X} \subset \mathbb{R}^m$ e delle rispettive assegnazioni alle classi di label $\{C_1, C_2\} = \{-1, +1\}$.
Un iperpiano in $\mathbb{R}^m$ si può definire come l'insieme

$$\{x : f(x) = w^\mathsf{T} x + b = 0\} \tag{3.26}$$

dove w rappresenta il versore ($\|w\| = 1$) ortogonale al piano.
La regola di classificazione può essere, dunque, fissata in maniera naturale come

$$\hat{y} = \text{sign}[w^\mathsf{T} x + b] \tag{3.27}$$

cioè come la distanza con segno tra il generico punto x e l'iperpiano (nota: manca la divisione per $\|w\|$ perché w è un versore).
Dato che, per ipotesi, le classi sono separabili, è possibile stimare i parametri della funzione $f(x)$ in modo tale che valga sempre

$$y_i f(x_i) > 0 \tag{3.28}$$

ossia in modo tale che l'iperpiano crei il massimo margine possibile tra le classi (v. figura 3.7). Questo si traduce nel problema di ottimizzazione

$$\begin{gathered} \max_{w,b,\|w\|=1} M \\ \text{s. a } y_i(w^\mathsf{T} x + b) \geq M,\ i = 1, \ldots, n \end{gathered} \tag{3.29}$$

Questo significa che la banda attorno all'iperpiano in cui non compaiono punti appartenenti ad alcuna classe è simmetrica e larga, in totale, $2M$.
Si noti che l'introduzione del parametro M è, in realtà superflua. In maniera più conveniente, si può considerare $M = \|w\|^{-1}$ e riformulare la 3.29 di conseguenza:

$$\begin{gathered} \min_{w,b} \|w\| \\ \text{s. a } y_i(w^\mathsf{T} x + b) \geq 1,\ i = 1, \ldots, n \end{gathered} \tag{3.30}$$

Poiché si utilizza un criterio di decisione quadratico e i vincoli sono lineari, tale problema di ottimizzazione risulta essere convesso.

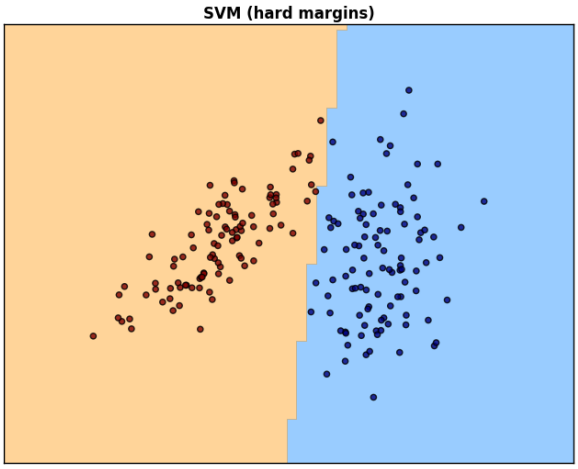


Figura 3.7: SVM con hard margins. Le regioni di decisione sono stimate a partire dai dati rappresentati. Il colore è indicativo della classe di appartenenza.

Tuttavia, il caso in cui le classi siano perfettamente separabili tramite un iperpiano è di scarso interesse, dato che, in primo luogo, si tratta di una situazione poco verosimile e, inoltre, quand'anche si verificasse, la soluzione non sarebbe sufficientemente robusta. Si deve, perciò, correggere per la possibile presenza di sovrapposizioni nel target space tra elementi appartenenti a classi diverse.
Un modo per fare questo è quello di massimizzare comunque la dimensione del margine, pur tuttavia ammettendo che alcuni punti si trovino dalla parte sbagliata rispetto a quest'ultimo (v. figura 3.8). Si introducono, quindi, n variabili di *slack* ξ_i, ciascuna delle quali indica di quanto il rispettivo punto x_i sfori il margine, in termini di frazioni di M.
Il problema 3.30 si può, allora, riformulare come

$$\begin{aligned} &\min_{w,b} \|w\| \\ \text{s. a } &\begin{cases} y_i(w^\top x + b) \geq 1 - \xi_i,\ i = 1, \dots, n \\ \xi_i \geq 0 \\ \sum_i \xi_i < h \end{cases} \end{aligned} \tag{3.31}$$

dove h è una costante che serve ad evitare di incappare nella soluzione banale per $\xi_i \to \infty$.
Si può, infine, riportare il tutto in una forma tipica della programmazione quadratica:

$$\begin{aligned} &\min_{w,b} \left\{ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \right\} \\ \text{s. a } &\begin{cases} y_i(w^\top x + b) \geq 1 - \xi_i,\ i = 1, \dots, n \\ \xi_i \geq 0 \end{cases} \end{aligned} \tag{3.32}$$

Il termine C, che sostituisce h, funge da fattore di regolarizzazione all'interno della funzione costo. Infatti, per $C \to \infty$ si ricade nel caso del problema 3.30 in

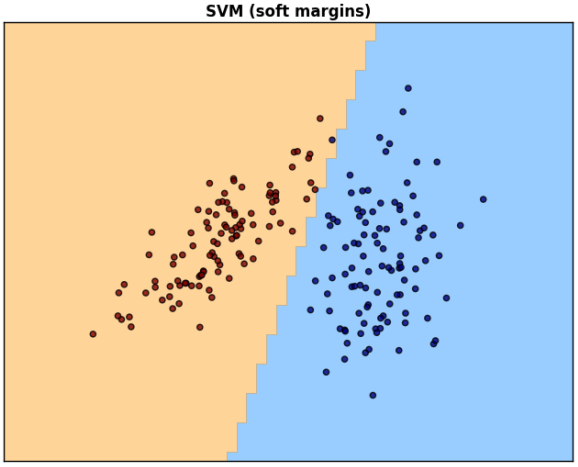


Figura 3.8: SVM con soft margins. Le regioni di decisione sono stimate a partire dai dati rappresentati. Il colore è indicativo della classe di appartenenza.

cui non sono ammessi sforamenti del margine. Invece, per $C \to 0$ tali sforamenti vengono sostanzialmente ignorati.
Si può notare, inoltre, che il valore di ciascuna variabile di slack ξ_i è indicativo della distanza (relativa) di ciascun punto dal margine dell'iperpiano appartenente al semispazio in cui dovrebbe risiedere la classe y_i. In particolare, la casistica per $\xi \geq 0$ (vedi vincoli) è la seguente:

- $\xi_i = 0$: il punto x_i si trova sul margine dell'iperpiano;
- $0 < \xi_i < 1$: il punto x_i si trova tra il margine e l'iperpiano, ma, comunque, nel semispazio corretto;
- $1 < \xi_i < 2$: il punto x_i si trova tra il margine e l'iperpiano, ma nel semispazio relativo alla classe opposta;
- $\xi_i > 2$: il punto x_i si trova oltre il margine dell'iperpiano nel semispazio sbagliato (grave errore di classficazione);

La 3.32 riportata sopra definisce il cosiddetto *primal problem* a cui corrisponde la forma duale

$$\max_{w,b} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^{\mathsf{T}} x_j \right\} \quad \text{s. a} \begin{cases} \sum_i \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases} \tag{3.33}$$

La derivazione della 3.33 viene omessa, così come l'elenco delle condizioni al contorno che caratterizzano l'unicità della soluzione. Infatti, più che il dettaglio matematico, è importante notare come, in questa forma, venga meno la dipendenza da w. In altre parole, la complessità computazionale della ricerca della soluzione ottima attraverso 3.33 non dipende dalle dimensioni del feature space

$\mathcal{X}$, ma solo dal numero n di dati a disposizione per la stima. Addirittura, tutti gli addendi si possono calcolare tramite prodotti scalari $x_i^\intercal x_j = \langle x_i, x_j \rangle$.
Un altro vantaggio è quello di ottenere una rappresentazione sparsa di

$$\hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i y_i x_i \tag{3.34}$$

dato che gli unici coefficienti $\hat{\alpha}_i$ non nulli sono quelli che corrispondono alle osservazioni x_i che cadono esattamente sul margine.

3.7.1 Kernel trick

Il principale svantaggio della SVM così come presentata in precedenza è la sua incapacità di risolvere problemi in cui la separazione tra le due classi sia non lineare.
Per far fronte a questo si può ricorrere ad un artifizio matematico che sfrutta una particolarità della forma duale 3.33 messa in evidenza qui sopra. Si parla, in tal caso, di *kernel trick*, poiché si sfruttano i cosiddetti kernel di Mercer, ossia trasformazioni del feature space tali per cui il prodotto scalare delle variabili trasformate sia esprimibile tramite semplici operazioni sulle variabili non trasformate.
Formalmente, una funzione $k(u, v)$ è un kernel di Mercer se, definita una trasformazione

$$\begin{aligned} \phi : \mathcal{X} &\to \mathcal{V} \\ x &\mapsto \phi(x) \end{aligned} \tag{3.35}$$

dal feature space $\mathcal{X}$ a uno spazio qualunque $\mathcal{V}$ in cui sia definito il prodotto scalare, si ha anche che

$$\begin{aligned} k : \mathcal{X} \times \mathcal{X} &\to \mathbb{R} \\ (u, v) &\mapsto \langle \phi(u), \phi(v) \rangle_{\mathcal{V}} \end{aligned} \tag{3.36}$$

Una volta scelta k e verficato che si tratti di un kernel di Mercer, è possibile inserirla direttamente all'interno della forma duale 3.33.
Così facendo, non è necessario calcolare esplicitamente la trasformazione ϕ (cosa, peraltro, non sempre possibile con un numero finito di termini) per ottenere un risultato, ma è sufficiente calcolare $k(x_i, x_j)$ in luogo del prodotto scalare $x_i^\intercal x_j = \langle x_i, x_j \rangle$:

$$\begin{gathered} \max_{w,b} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right\} \\ \text{s. a } \begin{cases} \sum_i \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases} \end{gathered} \tag{3.37}$$

L'utilità di tale risultato risiede nel fatto che, se i dati non sono linearmente separabili nel feature space originale $\mathcal{X}$, potrebbero esserlo in un appropriato spazio trasformato $\mathcal{V}$. Risolvere il problema 3.37 permette di verificare agevolmente se, in effetti, questo sia vero per un determinato kernel k.
Purtroppo, nonostante il kernel trick possa migliorare anche di molto le prestazioni del classificatore, porta con sé uno svantaggio potenzialmente molto grave:

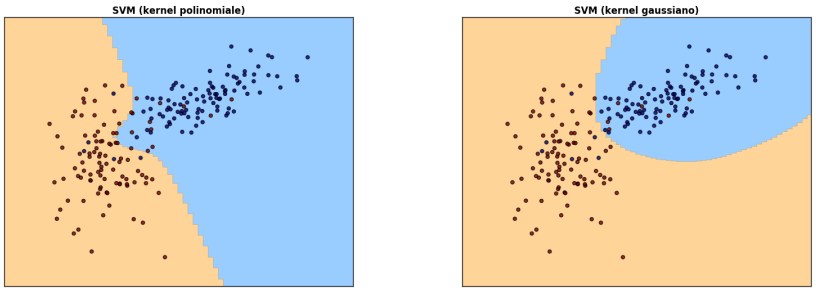


Figura 3.9: SVM con kernel polinomiale (a sinistra) e gaussiano (a destra). Le regioni di decisione sono stimate a partire dai dati rappresentati. Il colore è indicativo della classe di appartenenza.

una volta applicato fa perdere qualunque informazione relativa all'interpretatazione modellistica e probabilistica dei risultati.

Nonostante questo, nella maggior parte dei casi è più che sufficiente ottenere una *black box* attendibile e ivi risiede la fortuna delle SVM rispetto ad metodi parametrici.

Per concludere, si presentano alcuni esempi di kernel molto importanti e frequentemente utilizzati.

- Kernel lineare (caso limite in cui $\phi(x) = x$)

$$k(u, v) = \langle u, v \rangle$$

- Kernel polinomiale ($c_0 \in \mathbb{R}$, $d \in \mathbb{N}$)

$$k(u, v) = (\langle u, v \rangle + c_0)^d$$

- Kernel gaussiano o *radial basis function* ($\gamma \in \mathbb{R}^+$)

$$k(u, v) = e^{-\gamma \|u-v\|^2}$$

Per apprezzare come la scelta del kernel influenzi pesantemente i risultati si confrontino i due pannelli di figura 3.9.

3.8 Considerazioni riassuntive

In questo capitolo si sono prese in considerazione tecniche di classificazione molto diverse fra loro sia per complessità, sia per capacità di separazione fra i dati. Si è tentato, inoltre, di delineare un percorso logico per guidare nella scelta del clasificatore più adatto a una particolare esigenza, sulla base di considerazioni di carattere generale circa i punti deboli e di forza di ciascun metodo.

Si è partiti, dunque, dalla descrizione di semplici metodi di separazione lineare, tra loro concettualmente diversi: la tecnica basata sulla sogliatura della regressione permette di evidenziare la direzione di miglior approssimazione dei dati, il FLD quella di miglior separazione sotto ipotesi di gaussianità e la regressione logistica cerca di risolvere il problema degli outliers. Dopodiché, si è proposta

un'alternativa assolutamente non parametrica a queste tecniche, utile per seguire correttamente anche superfici di separazione fortemente non lineari, ma poco scalabile a dataset molto estesi.
Giunti a quel punto, si è presentata la dicotomia tra i metodi di ensemble, che combinano i risultati di molti classificatori semplici, e tecniche basate su modelli matematici più complessi come le SVM. Nel primo caso, per fornire una panoramica quanto più possibile completa, si sono delineate le caratteristiche di boosting, bagging e random forest. Invece, nel secondo, si è sviluppato il concetto di kernel trick che permette di convertire una tecnica basata sulla separabilità lineare tra le classi in uno dei metodi più efficaci e flessibili in circolazione.
Tuttavia, per portare a termine un protocollo di classificazione efficace, non è sufficiente scegliere un metodo e applicarlo ciecamente. È necessario, infatti, disporre di strumenti formali e pratici atti alla valutazione delle performance e, addirittura, all'individuazione delle feature più utili per la descrizione del problema in esame. Il capitolo successivo, dunque, presenta proprio queste problematiche, con particolare attenzione agli aspetti più subdoli della questione che, spesso, inducono in errore lo sperimentatore.

Capitolo 4

Metodologie per training, test, validazione e feature selection

4.1 Concetti di base

Una volta messo a punto un classificatore, il passo successivo è valutarne le prestazioni, in modo da verificare che sia effettivamente adatto al compito per il quale è stato progettato. Nell'ambito dell'apprendimento supervisionato, questo si traduce nell'applicare alcune definizioni in modo tale da ottenere una misura affidabile delle performance da diversi punti di vista complementari [35].
Prima di scendere nel dettaglio, è necessario introdurre due definizioni inerenti la suddivisione dei dataset: training e test set.
Per *training set* si intende l'insieme delle coppie $(x_i, y_i)_{i=1,\dots,n}$ che vengono utilizzate per la stima dei parametri o, comunque, per definire una regola di decisione (come nel caso del KNN descritto nella sezione 3.5).
Il *test set*, invece, è un ulteriore insieme di dati $(x_i, y_i)_{i=1,\dots,n_{test}}$ su cui un classificatore, già pronto all'uso, viene messo alla prova. Chiaramente, in entrambi i casi, una generica coppia (x, y) è formata, nell'ordine, da un elemento del feature space $\mathcal{X} \subset \mathbb{R}^m$ e dal corrispettivo nel target space $\mathcal{Y}$ (tale che $y \in \mathcal{Y}$ può assumere un unico valore tra $\{C_1, \dots, C_k\}$, nel caso d'interesse della classificazione).
In generale, training e test set vengono indicati con la stessa notazione con cui si designano le matrici formate dai loro elementi aggregati per righe. Per esempio, nel caso di quanto visto nella sezione 3.2, si potrebbero individuare X_{train}, Y_{train}, X_{test}, e Y_{test}, con ovvio significato dei simboli utilizzati. Si noti, inoltre, che il pedice può essere omesso se non c'è rischio di dare adito ad ambiguità.
Va da sé che training e test set debbano essere quanto più possibile indipendenti tra loro; al minimo, deve valere che $X_{train} \cap X_{test} = \varnothing$, cioè non devono esserci elementi in comune tra i due.

4.2 Training error

Il *training error* è la misura delle prestazioni di classificazione più intuitiva. Viene calcolata in maniera molto semplice applicando il classificatore al training set e sommando il numero di errori commessi. Generalmente, viene espresso in frazione o percentuale.
Volendo formalizzare il tutto, si può scrivere:

$$\varepsilon_{train} \coloneqq \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(y_i \neq \hat{y}_i) \tag{4.1}$$

Si ponga attenzione al fatto che ε_{train} può essere, spesso, spinto molto vicino a 0 se la stima dei parametri del classificatore avviene rimuovendo i termini di regolarizzazione dalla funzione costo. Si tratta della consueta considerazione sul compromesso bias-varianza, per cui, al diminuire dell'errore di ricostruzione dell'output desiderato, la varianza dello stimatore aumenta, potenzialmente fino a livelli inaccettabili.
Il training error, quindi, se esaminato da solo, è ben poco indicativo, a meno del caso raro e drammatico in cui sia particolarmente elevato, il che quasi sicuramente indica una forte inadeguatezza del classificatore in esame. Proprio per questo motivo, ε_{train} non viene mai utilizzato per aggiustare il valore di iperparametri quali i fattori di regolarizzazione.

4.3 Test error

L'impiego di un test set indipendente permette di ottenere una prospettiva differente rispetto a quella messa in luce dal training error.
La definizione di

$$\varepsilon_{test} \coloneqq \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \mathbb{I}(y_i \neq \hat{y}_i) \tag{4.2}$$

è formalmente identica a quella presentata in 4.1, ma è riferita al test set.
Il vantaggio di questa metrica risiede nel fatto che nessuno degli elementi del test set è stato utilizzato per la messa a punto del classificatore. Il suo valore, dunque, non può essere artificialmente abbassato agendo a livello della funzione costo.
Il rischio, tuttavia, è quello di incappare in una situazione particolare in cui, più per caso che per merito, le prestazioni risultino eccezionali. In casi simili, che sono ragionevolmente rari, ma sfortunatamente difficili da evitare, si parla di *overfitting* del test set.

4.4 K-fold cross-validation

Dalla combinazione di training e test error, in generale, è possibile ottenere una sintesi soddisfacente delle performance del classificatore.
Si tratta, tuttavia, di due valori puntuali, cioè dei quali non si conosce l'eventuale incertezza. In altre parole, non c'è alcuna garanzia che ottimi valori di ε_{train} e ε_{test} non siano semplicemente frutto di un colpo di fortuna.

In realtà, avere a disposizione un test set e un training set particolarmente numerosi è, all'atto pratico, di grande conforto circa l'attendibilità della stima degli errori. Spesso, però, specie nelle applicazioni in ambito biomedicale in cui acquisire un'osservazione può comportare disagi per il paziente, capita di poter contare su relativamente pochi dati.
Come si può intuire, in condizioni simili, la suddivisione del dataset in un training e un test set indipendenti è non solo poco indicata, ma, anzi, controproducente e dannosa. Dunque, è necessario trovare il modo di sfruttare completamente il training set nella fase di messa a punto del classificatore e, nel contempo, di ottenere informazioni simili a quelle ricavabili grazie a ε_{test}.
Tra le varie tecniche con cui ricavare un risultato che risponda a tale esigenza, quella di maggior successo è, senz'altro, la cosiddetta *K-fold cross-validation*, una procedura di validazione basata sulla suddivisione del training set in K parti. Di volta in volta, ciascuna delle K porzioni è utilizzata come test set, mentre le restanti $K-1$ sono coinvolte nella stima dei parametri o nella definizione della regola di decisione.
Dopo aver valutato $\varepsilon_{i,test}$ all'interno di ciascuna fold, si definisce l'errore di cross-validazione come la media

$$\varepsilon_{CV} \coloneqq \frac{1}{K}\sum_{i=1}^{K}\varepsilon_{i,test} \tag{4.3}$$

a cui si può affiancare, eventualmente, una misura di incertezza quale, per esempio, la deviazione standard inter-fold.
La scelta di K è completamente empirica e, ad oggi, non è stata ancora fornita una motivazione rigorosa in favore di una scelta o di un'altra. In ogni caso, alcuni valori comunemente utilizzati sono 4, 5 e 10, che sembrano dare risultati soddifacenti. Si sconsiglia, invece, di porre $K = n = |X_{train}|$ (*leave-one-out cross-validation*) poiché questo sembra dar luogo a stime troppo pessimistiche.

4.4.1 Interpretazione probabilistica

La definizione dell'errore di cross-validazione per sopperire alla presenza di un test error indipendente, così come presentata in precedenza, è, senza dubbio, ragionevole e convincente. Tuttavia, sembra opportuno accennare brevemente ad una sua intepretazione probabilistica, se non altro per dare l'idea dei solidi fondamenti teorici da cui trae origine.
Formalmente, si può definire il test error come

$$\mathrm{Err}_{\mathcal{T}} \coloneqq \mathrm{E}[L(Y,\hat{Y})|\mathcal{T}] \tag{4.4}$$

dove $L(Y,\hat{Y})$ è il valore di una generica funzione costo (per esempio il numero di errori) che funge da misura delle performance di classificazione, mentre la dipendenza da $\mathcal{T}$ sta ad indicare che si vuole stimare proprio l'errore relativo a un determinato test set. Evidentemente, nel caso non si avesse a disposizione tale test set indipendente, l'unico modo per risalire a questa informazione sarebbe conoscere esattamente che porzione dell'insieme universo dei dati sia rappresentata da $\mathcal{T}$. È chiaro, inoltre, che tale conoscenza renderebbe inutile la messa a punto di un classificatore.
Per eliminare la dipendenza da $\mathcal{T}$, quindi, si può calcolare il valore atteso di $\mathrm{Err}_{\mathcal{T}}$

rispetto a tutto quanto è aleatorio nella sua definizione, inclusa l'estrazione del training set:

$$\mathrm{Err} := \mathrm{E}[L(Y, \hat{Y})] = \mathrm{E}[\mathrm{Err}_{\mathcal{T}}] \tag{4.5}$$

A differenza dell'errore condizionato dell'equazione 4.4, Err può essere stimata agevolmente anche avendo a disposizione il solo training set tramite tecniche quali la cross-validazione.
Intuitivamente, la K-fold cross-validation approssima bene Err (ma non altrettanto bene $\mathrm{Err}_{\mathcal{T}}$!) a patto che il training set sia rappresentativo dell'universo dei dati: basta sostituire al valore atteso rispetto al test set la media artimetica tra le K fold.

4.4.2 Il problema del bias

Una questione di vitale importanza per la corretta valutazione e interpretazione delle misure di performance presentate in questo capitolo e, in particolar modo, della cross-validation riguarda l'introduzione artificiale di un (anche ingente) bias all'interno della stima dell'errore.
Il tutto scaturisce, in sostanza, dalla mancata osservanza (almeno una volta) dell'ipotesi di indipendenza tra training e test set all'interno dell'intera procedura di messa a punto e validazione del classificatore. Purtroppo, si tratta di un errore di non facile diagnosi. Da un lato, infatti, può sfuggire, sommerso dalla grande mole di operazioni, a volte complesse, necessarie per risolvere in maniera soddisfacente un problema di classificazione. Dall'altro, si tratta quasi sempre di un bias in senso ottimistico della cui presenza, quindi, si deve paradossalmente sospettare quando tutto sembra essere andato per il meglio.
Fortunatamente, però, nella quasi totalità dei casi, il bias viene ingenuamente introdotto in due momenti di facile individuazione all'interno delle procedure di costruzione e test del classificatore. Sarà, quindi, sufficiente che lo sperimentatore progetti con accortezza particolare quei due passi per non avere problemi di sorta.
Il primo momento critico è quello della scelta della regola di decisione che deve prescindere in modo assoluto dalla conoscenza di qualche pattern all'interno detest set (o del training set, nel caso della cross-validation). Infatti, così facendo, $\hat{Y}$ in 4.4, per esempio, risulterebbe essere, in realtà, una funzione di Y, ossia della classe assegnazione alle classi, in grave violazione dell'indipendeza tra training e test set. Situazioni simili sono frequenti nel caso in cui si voglia cercare di introdurre della conoscenza pregressa nel classificatore per facilitarne il compito. In questi casi, bisogna essere in grado di isolare abilmente le informazioni che derivano da constatazioni di carattere del tutto generale (che sono ammesse) da quelle ottenute spiando le caratteristiche del test set.
In parole povere, con un esempio banalissimo, è lecito sfruttare la conoscenza che le ciliegie, in genere, siano più piccole dei pomodori, ma non è pensabile prendere una decisione sulla base del fatto che queste sono rosse, mentre quelli sono gialli solo perché si conoscono le particolari varietà che costituiscono il test set.
Il secondo momento in cui bisogna ben riflettere è quello dell'impostazione della procedura di cross-validazione. L'ipotesi di indipendenza tra training e test set, infatti, continua a dover essere rispettata nonostante, in realtà, sia la porzione di test, sia quella di training siano state ricavate suddividendo il training set

iniziale in K fold. Per esempio, quindi, se si ritenesse utile normalizzare i dati sottraendo la loro media e la loro varianza, tale media e tale varianza dovrebbero essere stimate K volte, nelle K porzioni di training, prima di essere utilizzate sulle corrette porzioni di test.
Allo stessso modo, qualunque procedura di ottimizzazione che si basi un'ulteriore cross-validazione deve essere effettuata dividendo, per ogni K, la sola porzione di training di volta in volta considerata in K' parti su cui applicare la nuova cross-validation.
In sintesi, bisogna sempre assicurarsi che non accada mai che $\hat{Y}^\tau$ sia funzione di X^τ_{test} o Y^τ_{test} per ogni step τ dal momento della predisposizione di una regola di decisione fino all'ultima validazione del classificatore.

4.5 Feature selection

Una domanda che ha senso porsi quando si è messi di fronte a un problema di classificazione è se, in realtà, sia necessario utilizzare tutte le informazioni che si hanno a disposizione. Se, da un lato, utilizzare quante più osservazioni possibile è, senz'altro, un approccio ragionevole, dall'altro è abbastanza facile immaginare innumerevoli situazioni in cui non tutte le caratteristiche dati siano necessarie per assegnarli a una classe. Potrebbe capitare, infatti, che una variabile sia sempre uguale per ogni elemento x_i del training set (numero di madri di un mammifero) oppure che sia un fattore confondente rispetto al problema in esame (ascendente zodiacale).
Si vorrebbe, quindi, trovare un modo per limitare il modello su cui si basa il classificatore al sottinsieme di feature più appropriato rispetto a una qualche metrica.
Nel seguito si presentano alcuni esempi di metodi (automatici e non) atti ad operare tale riduzione di dimensionalità del training set che va sotto il nome di *feature selection* [35].

4.5.1 Metodi ricorsivi

La ricerca del miglior sottinsieme di features rispetto a una data funzione costo difficilmente può essere portata a termine in maniera esaustiva esaminando ciascuna delle $\sum_{k=1}^{m} \binom{m}{k}$ possibili combinazioni. Sono stati, quindi, proposti diversi algoritmi per approssimare il risultato del metodo *best subset*. Di seguito se ne presentano due casi emblematici che, nonostante siano le tecniche più semplici, trovano ancora amplissimo impiego nel settore.
Prima di descriverle, si ritiene utile riportare l'attenzione a quanto esposto nel paragrafo 4.4.2 circa il trasferimento di informazione tra training e test set: nel caso in cui la funzione costo fosse riferita a una parte teoricamente indipendente dei dati (esempio: errore di cross-validazione), tale condizione di indipendenza deve essere rispettata.

4.5.1.1 Forward Stepwise Selection

La forward stepwise selection è un algoritmo *greedy* attraverso cui l'insieme delle features selezionate, il cosiddetto *active set*, S viene arricchito di una variabile

ad ogni iterazione, quella che dà luogo al massimo miglioramento della funzione costo. La speranza è che il sottinsieme creato in questo modo approssimi adeguatamente il *best subset*, nonostante non ci sia alcuna garanzia che se anche S^t fosse effettivamente il best subset di t feature alla t-esima iterazione, $S^{t+1} = S^t \cup \{F\}$ rimanga il sottinsieme ottimo.
L'algoritmo procede, quindi, come segue.

1. Inizializzare l'active set a $S = \varnothing$.
2. Fino a che S non contiene il numero di feature desiderate (o altra condizione equivalente):
   (a) Per ciascuna delle feature F_i ancora non inserite in S:
      i. Mettere a punto il classificatore sulle feature $S \cup \{F_i\}$.
      ii. Valutare la funzione costo.
   (b) Individuare la feature $F = F_i$ dalla cui aggiunta si ottiene il massimo miglioramento della funzione costo.
   (c) Aggiornare $S = S \cup \{F\}$, aggiungendo F.
3. Dare in output l'active set finale S

4.5.1.2 Backwards Stepwise Selection

Agli antipodi rispetto alla forward stepwise selection, la backwards stepwise selection prevede di eliminare ricorsivamente dall'active set la feature che risulta meno utile dal punto di vista della funzione costo.
L'algoritmo procede in maniera speculare a quello presentato nella sezione precedente.

1. Inizializzare l'active set a $S = \{F_1, \ldots, F_m\}$, inserendo tutte le feature.
2. Fino a che S non contiene il numero di feature desiderate (o altra condizione equivalente):
   (a) Per ciascuna delle feature F_i ancora presenti in S:
      i. Mettere a punto il classificatore sulle feature $S - \{F_i\}$.
      ii. Valutare la funzione costo.
   (b) Individuare la feature $F = F_i$ dalla cui eliminazione si ottiene il massimo peggioramento della funzione costo.
   (c) Aggiornare $S = S - \{F\}$, eliminando F.
3. Dare in output l'active set finale S

Un caso particolare di questa tecnica che va sotto il nome di Recursive Feature Elimination mira a diminuire la complessità computazionale dell'algoritmo di partenza eliminando gruppi di variabili (anziché una sola feature) sulla base della loro importanza [36].

4.5.2 Penalizzazione L1

Una categoria di metodi alternativi a quelli ricorsivi per la ricerca di un sottinsieme ottimo di feature si basa sulla cosiddetta penalizzazione L1, ossia sull'inserimento (o la modifica) di un termine di regolarizzazione all'interno della funzione costo per la stima dei parametri, con l'ovvio vantaggio di poter effettuare entrambe le operazioni (stima e selezione) contemporaneamente.
Il metodo, per ragioni che saranno immediatamente ovvie, è limitato a classificatori parametrici, cioè caraterizzati dalla necessità di stimare un weight vector finito e di dimensione indipendente da quella dei dati. Infatti, è applicabile in maniera del tutto generale a qualunque funzione costo purché si introduca una qualche dipendenza da

$$\|w\|_1 = \sum_j |w_j| \tag{4.6}$$

ossia dalla somma dei valori assoluti delle componenti di w. In particolare, data una funzione da ottimizzare $L(Y, \hat{Y})$ la sua versione penalizzata L1 si può quasi sempre ottenere in due modi, a seconda che la sua forma originale dipenda già da $\|w\|$ oppure no.
Se $L(Y, \hat{Y})$ non è funzione di $\|w\|$, basta aggiungere un termine contenente la norma 1 dell'equazione 4.6, in genere moltiplicato per una costante:

$$L_{L1}(Y, \hat{Y}) \coloneqq L(Y, \hat{Y}) + \gamma \|w\|_1 \tag{4.7}$$

Altrimenti, se nella formulazione originale di $L(Y, \hat{Y})$ si può isolare un termine che è funzione della norma di w, cioè

$$L(Y, \hat{Y}) = L_0(Y, \hat{Y}) + f(\|w\|)$$

tale termine deve essere rimosso per far posto alla dipendenza dalla norma 1:

$$\begin{aligned} L_{L1}(Y, \hat{Y}) &\coloneqq L(Y, \hat{Y}) - f(\|w\|) + \gamma \|w\|_1 = \\ &= L_0(Y, \hat{Y}) + \gamma \|w\|_1 \end{aligned} \tag{4.8}$$

4.5.3 Expert Knowledge

In maniera preliminare rispetto all'applicazione dei metodi automatici di cui si sono visti alcuni esempi (o proprio perché questi sembrano fallire) la feature selection può anche essere effettuata a priori introducendo quella che, in gergo, si definisce *expert knowledge*.
Semplicemente, prima di approntare il classificatore, si definisce un insieme di feature che si suppone siano la combinazione migliore per risolvere il problema di classificazione. Tuttavia, sempre alla luce delle considerazioni fatte nella sezione 4.4.2, devono essere rispettati alcuni vincoli in modo da evitare lo scambio di informazioni tra training e test set. In particolare, questo significa che la scelta delle variabili da mantenere non deve essere fatta confrontando la loro variazione tra le classi a meno che non si escludano dal test set (o dal training set, in caso di cross-validation) le osservazioni utilizzate per la selezione. È possibile, invece, considerare tranquillamente tutti i dati che si desidera se le conclusioni sulla bontà delle feature vengono tratte senza spiare in alcun modo le etichette.
In questo senso (lato), si possono includere nelle tecniche che si basano sulla expert knowledge anche procedure automatiche, quali la *Principal Component*

Analysis, purché, nel momento in cui vengono applicate, non siano in grado di accedere ad alcuna informazione relativa alle label.
In parole povere, non si deve scadere nel grossolano errore di scegliere le feature suddividendo, direttamente o indirettamente, i dati in classi, pena la nullità dell'intero procedimento di analisi delle prestazioni che, così facendo, con alta probabilità, risulterebbero affette da un bias in senso ottimistico.

Capitolo 5

Database e setup delle metodologie

5.1 Database

I soggetti a disposizione per questo lavoro sono stati individuati nell'ambito del progetto Mosaic [37]. Si tratta, in particolare, di una esigua sottopopolazione estratta da due importanti studi a lungo termine, il Botnia Perspective Study (BPS) e il Botnia PPP Study [38][39][40].
I pazienti iseriti nel database rispondono ai seguenti criteri di inclusione:

- diagnosi univoca tramite gold standard (v. sezione 1.4) di IGT o T2D a meno di tre anni di distanza dalla loro partecipazione in uno dei due studi;
- età compresa tra i 40 e 75 anni;
- nessuna terapia in corso con insulina, analoghi GLP-1, sulfaniluee, corticosteroidi ad assunzione orale, agenti tireostatici or ormoni tiroidei, analoghi LHRH;
- non in gravidanza;
- nessun cambiamento noto nelle fotografie del fondo della retina;
- assenza di microalbuminuria;
- HbA1c $< 8\%$;
- FPG > 180 mg/dl;
- interruzione di un eventuale trattamento con gliptine o acarbosio almeno due giorni prima del test e per tutta la durata dello studio.

Il database finale utilizzato si compone di 62 pazienti per ognuno dei quali si hanno a disposizione un tracciato CGM acquisito con tempo di campionamento di 5 minuti per alcuni giorni (rarissime le interruzioni) e una lunga serie di ulteriori parametri clinici, tra cui quelli presentati nel paragrafo 2.10. Queste informazioni si riferiscono sia alla visita iniziale avvenuta per ciascun paziente tra febbraio e giugno 2014, sia alla seconda visita del periodo marzo-giugno

2015, in occasione della quale sono stati nuovamente acquisiti.
In occasione della visita 1, 36 pazienti erano affetti da IGT e 26 da T2D mentre, un anno dopo, le due categorie contavano, rispettivamente, 37 e 25 pazienti in seguito a due miglioramenti da T2D a IGT e un peggioramento da IGT a T2D.

5.2 Training e test set

Per poter applicare le tecniche di classificazione viste nel capitolo 3, si è suddiviso il database di partenza in modo molto naturale, considerando come training set i dati relativi alla prima visita e come test set quelli relativi alla seconda.
Chiaramente, tra i due set si rileva una certa correlazione, in quanto si tratta degli stessi pazienti visti a un anno di distanza, tuttavia si è preferito mantenere il raggruppamento in termini cronologici per ottenere sottinsiemi quanto più omogenei possibile rispetto all'acquisizione. Nonostante questo, premesse e risultati sono piuttosto confortanti in tal senso. Infatti, innanzi tutto, la diagnosi può cambiare tra una visita e l'altra e, inoltre, come si vedrà, non si rileva nessuna tendenza dominante nella classificazione a riassegnare ciascun soggetto alla sua classe originale.

5.3 Scelta delle classi

Il problema di classificazione affrontato in questo lavoro è quello della distinzione tra i soggetti affetti da IGT e quelli per cui si può parlare a tutti gli effetti di T2D.
Per ogni paziente inserito nel database, come si diceva nella sezione 5.1, è disponibile una diagnosi perfettamente attendibile, ottenuta tramite gold standard. Si sono, quindi, codificate due classi nel modo seguente:

- un soggetto appartiene alla classe $C_1 = 1$ se affetto da IGT;
- un soggetto appartiene alla classe $C_2 = 2$ se affetto da T2D.

Le etichette delle classi sono passate, in fase di classificazione, come i due numeri interi 1 e 2 sia in ingresso, sia in uscita. Qualora il classificatore utilizzato richieda una diversa codifica, ci si adegua alle indicazioni presentate nel capitolo 3.

5.3.1 Ragioni della scelta

A questo punto, si ritiene opportuno delineare il razionale che ha spinto alla selezione, tra i tanti, proprio del problema "IGT contro T2D". Si potrebbe obiettare, infatti, che non si tratti della questione più interessante da dirimere, vista l'abbondanza di metodi gold standard a disposizione.
È probabilmente per questo motivo, infatti, che, in letteratura, tecniche di classificazione basate sulla variabilità glicemica vengono applicate più di frequente ai soggetti affetti da diabete di tipo 1 [24][25][26][30]. L'obiettivo, in genere, è quello di caratterizzare il livello di gravità della patologia sulla base del fatto il paziente sia più o meno strettamente controllato, ovvero se la frequenza e gravità degli episodi di ipoglicemia e iperglicemia sia accettabile oppure no.
L'approccio prediletto in questo senso sembra essere quello della definizione,

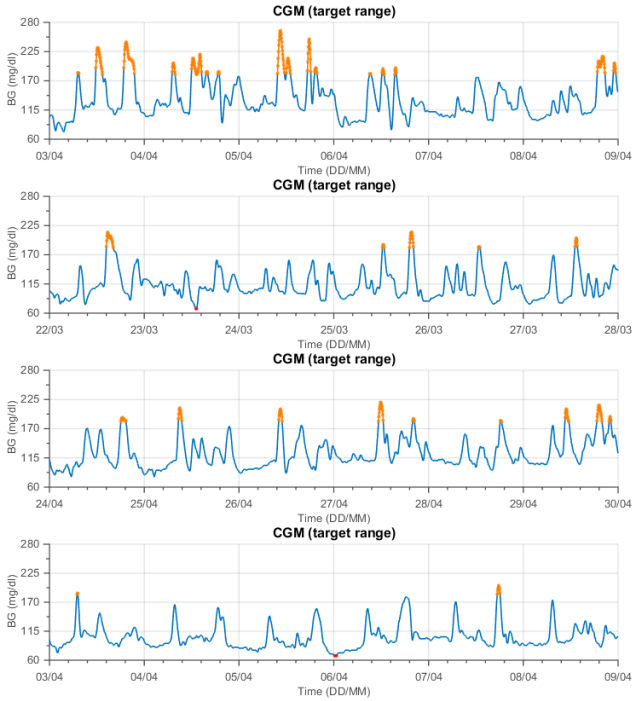


Figura 5.1: Esempi di quattro tracciati CGM, due di pazienti IGT e due di pazienti T2D.

da parte di un gruppo di esperti, di una suddivisione arbitraria in classi di rischio sulla base di un'ispezione visiva del solo tracciato CGM corredato da informazioni circa l'entità e la cadenza dei pasti. La procedura automatizzata di classificazione tramite metodi simili a quelli presentati nel capitolo 3 viene, poi, portata avanti sulla base della ripartizione effettuata.
Apparentemente, si tratta di un'assunzione molto ragionevole, tuttavia nessuna delle tecniche applicate nei lavori esaminati sembra tener conto dell'incertezza che, così facendo, si introduce sulle classi. Per rendersi conto dell'entità del problema, si riporta, traducendolo quanto più letteralmente possibile, un passo tratto da [24]:

> *Nel corso di questo test [classificare 100 tracciati CGM, ndr] la coerenzia media intra-medico è stata 82%, con un medico che ha classificato coerentemente 81 su 100 tracciati e l'altro 83 su 100. Per quanto riguarda i 67 tracciati su cui entrambi i medici sono stati internamente coerenti, i due si sono trovati in accordo 61 volte, per un consenso intra-medico del 91%.*

Evidentemente, un tasso di incertezza simile richiederebbe un ulteriore passo di correzione o, almeno, un affinamento della tecnica di valutazione.
Un secondo ostacolo all'applicazione degli indici di variabilità glicemica per problemi non ben codificati può essere rappresentato dalla scarsa numerosità dei dati. Infatti, l'estrazione manuale di un *pattern* attraverso cui operare la sud-

divisione iniziale in classi può avere senso soltanto a fronte di database molto corposi.
Infine, si deve tener conto dello stato embrionale dell'applicazione estensiva della variabilità glicemica per questo genere di applicazioni.
Dunque, tenendo conto di queste considerazioni, si è ritenuto opportuno perseguire l'obiettivo meno ambizioso di riprodurre i risultati di test diagnostici di livello gold standard tramite l'analisi del segnale CGM. Tuttavia, sebbene la distinzione tra IGT e T2D sia perfettamente codificata a livello di HbA1c e affini, i segnali CGM non mostrano alcuna apparente correlazione con la reale condizione del paziente a fronte di una semplice ispezione visiva.
Per convincersene, in figura 5.1 sono rappresentati quattro segnali, due per classe. Prima di procedere nella lettura, si provi a ipotizzare, sulla scorta di quanto esposto nel capitolo 1 quali siano i due pazienti affetti dalla condizione più grave, il diabete di tipo 2.
Probabilmente in modo inaspettato, si tratta del primo (prevedibile) e, sorprendentemente, dell'ultimo il che è di conforto circa la non banalità del problema.

5.4 Scenari di classificazione

Come si è visto, un passo importante nella scelta del classificatore più opportuno per una data applicazione è la selezione delle feature.
In questo lavoro si sono presi in considerazione tre scenari di base, caratterizzati da un livello di conoscenza dello stato di salute del soggetto via via crescente.

- Scenario 0: sono a disposizione solo i dati di sensore da cui si estraggono tutti gli indici di variabilità glicemica presentati nel capitolo 2.
- Scenario 1: si ipotizza di conoscere anche i parametri clinici di base il cui costo di acquisizione, come discusso nella sezione 2.10, può considerarsi nullo.
- Scenario 2: oltre a quanto incluso negli scenari precedenti, si suppone di aver sottoposto il paziente ad alcuni esami aggiuntivi per ottenere i valori elencati nella sezione 2.10.

Nello specifico, le m variabili considerate sono riportate sotto.

- Scenario 0 ($m = 37$): media, SD, CV, mediana, range, IQR, J-index, SDw, SDm, $\%\text{BG}_{within}$, $\%\text{BG}_{above}$, $\%\text{BG}_{below}$, M_{100}, Hypo Index, Hyper Index, IGC, GRADE, GRADE_{eu}, GRADE_{hypo}, GRADE_{hyper}, MAGE, MAGE+, MAGE-, EF, LBGI, HBGI, BGRI, ADRR, CONGA_4, MODD, ϕ_1, ϕ_2, ϕ_3, ϕ_4, ϕ_5, ϕ_6, ϕ_7.
- Scenario 1 ($m = 41$): sesso, età, BMI, circonferenza della vita (oltre alle precedenti).
- Scenario 2 ($m = 45$): pressione sistolica, pressione diastolica, frequenza del battito cardiaco, fasting glucose (oltre alle precedenti).

Per i classificatori più promettenti, poi, è stato messo a punto un ulteriore scenario di classificazione basato sulla feature selection manuale, tramite expert knowledge, effettuata grazie al supporto di un esperto diabetologo.

La scelta è avvenuta in maniera molto semplice, mostrando al clinico un riepilogo degli indici e come essi si applicano a un tracciato CGM scelto casualmente di cui nessuno dei presenti conosceva la classe di appartenenza.
L'esclusione delle feature è avvenuta sulla base di due criteri, etrambi indipendenti dalla scelta delle classi:

1. scarsa interpretabilità dal punto di vista clinico o, comunque, contributo informativo poco attinente alla patologia del diabete;

2. ridondanza (come nel caso delle %BG che sommano a 100%).

Lo scenario conclusivo messo a punto in questo modo è riportato di seguito.

- Scenario 3 ($m = 22$): sesso, età, BMI, circonferenza della vita, fasting glucose, media, SD, CV, mediana, range, IQR, $\%BG_{within}$, $\%BG_{below}$, Hypo Index, Hyper Index, $GRADE_{hypo}$, $GRADE_{hyper}$, MAGE+, MAGE-, EF, LBGI, HBGI.

Si noti il circa dimezzamento della dimensionalità dei dati.

5.4.1 Aknowledgment

Si ringrazia il dott. Alberto Maran del Dipartimento di Medicina (DIMED) dell'Università degli Studi di Padova per il supporto alla creazione dello scenario 3.

5.5 Schema di classificazione

Per ognuno dei classificatori utilizzati, i risultati si riferiscono a una procedura in tre passi che, volendo individuarne lo schema generale, si può descrivere come segue:

1. standardizzazione del dataset (ogni variabile viene centrata e riscalata affinché nella popolazione abbia media nulla e varianza unitaria);

2. ricerca esaustiva degli iperparametri con il metodo *grid search*, utilizzando come funzione costo l'errore di cross-validazione;

3. utilizzo nel classificatore finale della combinazione di parametri e iperparametri corrispondente al minimo di tale funzione costo.

5.6 Implementazione della cross-validazione

L'errore di cross-validazione (v. sezione 4.4) è una delle misure di performance utilizzate nel seguito. Tuttavia, come si è visto nella sezione 4.4.2, la stima di tale metrica deve essere portata a termine con grande attenzione al mantenimento dell'indipendenza tra training e test set.
Dunque, con il duplice obiettivo di fugare ogni dubbio circa l'accuratezza dei risultati e di offrire un esempio di good practice per calcoli di questo tipo, si descrive per punti la procedura utilizzata per ricavare l'errore di cross-validazione. Chiaramente, ci si riferisce allo schema di classificazione esposto poco sopra.

1. Suddivere il training set X in K parti contenenti ciascuna lo stesso numero di elementi, estratti a caso da entrambe le classi (*stratified* K-fold cross-validation).
2. Per ciascuna delle $i = 1, \ldots, K$ fold:
   (a) Mettere da parte l'i-esima fold per utilizzarla come test set.
   (b) Condensare le altre $K - 1$ in un training set provvisorio.
   (c) Effettuare la grid search su una griglia di $c = 1, \ldots, C$ combinazioni di iperparametri.
      i. Suddividere l'unione delle $K - 1$ fold in ulteriori K' fold.
      ii. Per ciascuna delle $j = 1, \ldots, K'$ nuove fold:
         A. Mettere da parte la j-esima fold per utilizzarla come test set.
         B. Condensare le altre $K' - 1$ in un ulteriore training set provvisorio.
         C. Stimare media e deviazione standard solo sul training set provvisorio.
         D. Standardizzare training e test set provvisori sottraendo ad entrambi tale media e tale SD (che non dipendono dal contenuto della j-esima fold!).
         E. Stimare i parametri del classificatore sull'unione delle $K' - 1$ fold.
         F. Utilizzare il classificatore così ottenuto sulla j-esima fold (di test).
         G. Memorizzare l'errore di classificazione così ottenuto ε_c, j
      iii. Mediare $\varepsilon_{c,j}$ rispetto a j per ottenere l'errore di cross-validazione per una data combinazione di iperparametri c, ε_c.
   (d) Scegliere la combinazione di parametri c^* come $\text{argmin}_c\{\varepsilon_c\}$, cioè quella per cui è minimo l'errore di cross-validazione.
   (e) Stimare media e deviazione standard solo sulle $K-1$ fold di training.
   (f) Standardizzare il training set completo X sottraendo tale media e dividendo per tale deviazione standard (che non dipendono dal contenuto della i-esima fold!).
   (g) Fissata la combinazione c^*, stimare i parametri del classificatore sull'unione delle $K - 1$ fold.
   (h) Utilizzare il classificatore così ottenuto sull'i-esima fold (di test).
   (i) Memorizzare l'errore di classificazione err_i.
3. Calcolare la media inter-fold dell'insieme di errori err_i.
4. Tale media è l'errore di cross-validazione.

5.7 Software e hardware utilizzati

Per il calcolo degli indici, l'elaborazione dei tracciati e l'estrazione dei parametri clinici da file `.xlsx` si è utilizzato `MATLAB R2014b`.

Le procedure di classificazione sono state implementate grazie alla libreria Python `scikit-learn` versione `0.17.1` [41].
Il tutto è stato effettuato su un'unica macchina, in ambiente Windows 10 Pro 64-bit, dotata di processore Intel Core i5-6600 e 8 GB di RAM DDR4.
Tutti i random number generator sono stati inizializzati con il seed `42`.
In appendice B è possibile prendere visione del listato integrale delle sole *function* realizzate per il calcolo degli indici di variabilità glicemica. Inoltre, l'appendice C riporta una traccia del codice utilizzato, in generale, per ottenere i risultati presentati nel seguito.

Capitolo 6

Risultati

6.1 Premessa e criteri di interpretazione

Dalla letteratura, l'impressione è che problemi di classificazione simili a quello in esame (v. capitolo 5) vengano spesso affrontati procedendo per tentativi, ossia senza motivare la scelta di una tipologia di metodi o di un'altra [24][25][26][30]. Dunque, in questo lavoro ci si propone di mettere in evidenza una sorta di traiettoria logica che è possibile seguire per approfondire l'analisi. Nello specifico, si partirà con lo stabilire una *baseline* di prestazioni utilizzando le tecniche più semplici di classificazione lineare, e, laddove le performance non siano soddisfacenti, si ricorrerà a metodi incrementalmente più complessi. Per quanto possibile, si motiverà anche la scelta effettuata formulando una *educated guess* circa le ragioni del deterioramento delle performance e ricorrendo al classificatore che permetta, in virtù delle sue proprietà, di confermare o escludere la veridicità di tale ipotesi.
Nel seguito, quando lo si riterrà utile, saranno fornite per ciascun classificatore anche delle informazioni riassuntive costituite da:

- una rappresentazione pittorica nello spazio bidimensionale dei risultati della classificazione sul test set;
- una tabella con l'errore percentuale di training, test e cross-validazione;
- la matrice di confusione relativa al test set.

Per interpretare l'immagine, si tenga conto che essa è stata ottenuta proiettando tramite *Principal Component Analysis* [35] i dati in due dimensioni e, successivamente, equalizzando l'istogramma delle coordinate dei punti trasformati [42]. Va da sé che la rappresentazione è poco significativa dal punto di vista matematico, ma, nonostante ciò, può essere utile per avere un riscontro visivo su quali siano, nello specifico, le osservazioni correttamente classificate.
I simboli utilizzati nell'immagine sono, intuitivamente:

- pallino pieno in caso di classificazione corretta;
- croce in presenza di un errore di classificazione;
- colore blu (`#0072BD`) se IGT è la classe reale di appartenenza;

- colore arancio (`#D95319`) se T2D è la classe reale di appartenenza.

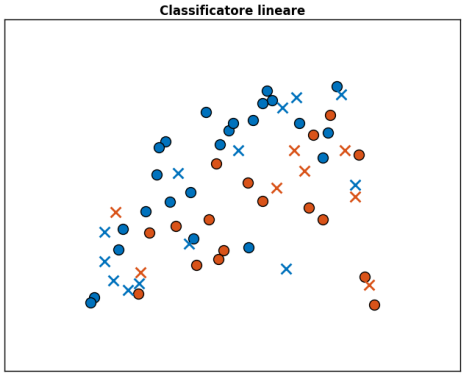
L'intera galleria dei risultati è disponibile in appendice A.

6.2 Baseline: classificazione lineare

Le tecniche di classificazione lineare, come si è visto, sono basate sulla separazione del target space in due semispazi, ciascuno occupato dagli elementi di una delle due classi. Si tratta dell'approccio più semplice al problema e, naturalmente, anche di quello che ha minori probabilità di fornire risultati soddisfacenti. Tuttavia, nell'ottica di cercare la metodologia più parsimoniosa per ottenere le migliori performance, si tratta di un passo obbligato, se non altro per escludere la lineare separabilità tra le classi.
A questo scopo, si sono messi a punto tre tipi di classificatori diversi, ciascuno dei quali di diversa utilità nell'individuazione dell'iperpiano separatore a seconda della distribuzione dei dati.

6.2.1 Classificatore lineare con indicator matrix coding

Il classificatore lineare basato sulla sogliatura di una regressione lineare su etichette monodimensionali, come visto nel capitolo 3, non è un candidato soddisfacente per la classificazione. Infatti, i risultati dipendono troppo dal criterio di sogliatura e dalla definizione delle label stesse. Per ovviare a questo problema, si è ricorso all'indicator matrix coding, ottenendo così una buona approssimazione del miglior risultato ottenibile attraverso un passo intermedio di regressione.
Come si può osservare, per esempio, dalla figura 7.1 (riferita allo scenario 1) buona parte degli elementi del test set viene classificata scorrettamente da questa tecnica elementare: circa un terzo dei pazienti affetti da T2D sono attribuiti erroneamente alla classe IGT e, contemporaneamente, a 13 su 24 soggetti prediabetici viene diagnosticato un inesistente T2D. Poiché considerazioni analoghe potrebbero essere fatte anche per gli altri scenari (v. appendice A), si può, dunque, affermare che non emerga alcun tipo di separabilità lineare tra le classi, ovvero che non sia apparentemente possibile sfruttare un iperpiano per dividere efficacemente il target space.



Scenario 1	
$\varepsilon_{\mathbf{train}}$	3.2%
$\varepsilon_{\mathbf{CV}}$	24.4%
$\varepsilon_{\mathbf{test}}$	33.9%

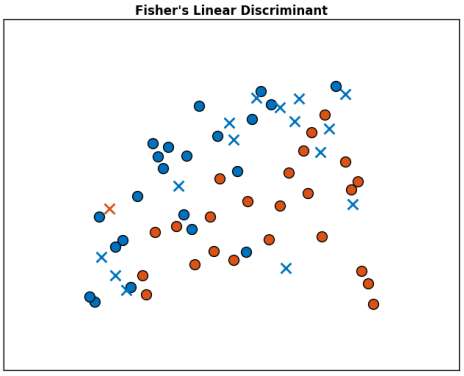
Realtà \ Predizione	IGT	T2D
IGT	24	13
T2D	8	17

Figura 7.1: classificatore lineare con indicator matrix coding applicato allo scenario 1.

Si faccia attenzione, poi, a non farsi ingannare dall'entità dell'errore che, sia per quanto riguarda la cross-validazione, sia sul test set rimane nell'ordine del 30%. Infatti, si ricorda che, sebbene "accuratezza del 70%" abbia un suono allettante, tale percentuale deve essere confrontata, da un lato, con la classificazione tramite lancio di una moneta che, in media, sbaglia una volta su due e, dall'altro, con l'attribuzione banale di tutte le osservazioni alla classe più numerosa il che, in questo caso, porterebbe ad accuratezza del circa 58%.
Sulla scorta di questi risultati, si può, quindi, ribadire di non aver ritrovato alcuna separabilità lineare. Tuttavia, per così dire, non tutto è perduto, poiché esistono, come si è visto, altri metodi, complementari a questo, che potrebbero avere esito più favorevole.

6.2.2 Fisher's Linear Discriminant

I risultati sconfortanti del semplice classificatore lineare basato sulla regressione e arricchito con indicator matrix coding potrebbero essere attribuibili al fatto che si tratti di un approccio poco appropriato alla distribuzione dei dati. Infatti, non c'è nessuna garanzia che la direzione di miglior approssimazione sia anche perpendicolare a quella di miglior separazione né che la sogliatura ottenuta sia, in effetti, la migliore possibile. Per questo, si è applicato il Fisher's Linear Discriminant che, appunto, utilizza come metrica alla base della stima proprio la bontà della separazione.
Come si evince dai risultati ottenuti, però, nonostante si possa rilevare un leggero miglioramento delle performance, che si fa un po' più marcato nel caso dello scenario 2 (quello più completo), si può, comunque, confermare la non separabilità lineare tra le classi. In particolare, ponendo l'attenzione proprio sull'ultimo caso, più favorevole, è facile notare che la buona accuratezza che appare sulla carta sia dovuta, più che altro, all'assegnazione incontrollata delle osservazioni alla classe T2D. Infatti, dall'analisi della matrice di confusione riportata in figura 7.2 traspare che, sebbene quasi non si registrino errori per quanto riguarda la corretta identificazione del diabete di tipo 2 (24/25 assegnazioni corrette), lo stesso non si possa assolutamente dire per la diagnosi di IGT che risulta, spesso (15/37 volte), inappropriata.



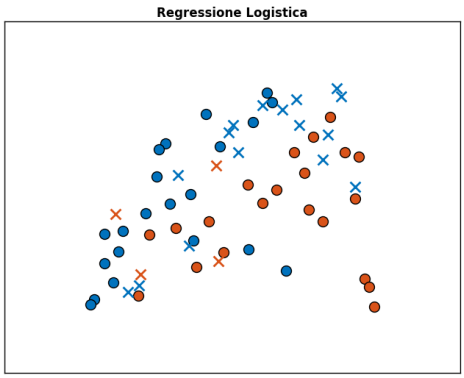
Scenario 2	
$\varepsilon_{\mathbf{train}}$	11.3%
$\varepsilon_{\mathbf{CV}}$	27.6%
$\varepsilon_{\mathbf{test}}$	25.8%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	22	15
	T2D	1	24

Figura 7.2: Fisher's Linear Discriminant applicato allo scenario 2.

6.2.3 Regressione logistica

Sebbene finora tutti gli elementi a disposizione puntino verso l'assenza di separabilità lineare, ha senso contemplare anche la possibilità che si tratti di un effetto collaterale della presenza di forti outlier. In tal caso, infatti, l'iperpiano separatore ottenuto con la sogliatura della regressione sarebbe traslato eccessivamente verso gli outliers stessi e, contestualmente, verrebbe meno l'ipotesi di gaussianità necessaria per la corretta applicazione del FLD.
Fortunatamente, la regressione logistica è ben adatta ad affrontare e risolvere questo problema. In particolare, si è applicata una variante della tecnica presentata nel capitolo 3 per la quale si è stimato un iperparametro di penalizzazione della norma quadratica del weight vector w scandendo una griglia (monodimensionale) di 200 valori del termine di regolarizzazione C nell'intervallo $[1, 99]$.
Nonostante tutti questi sforzi, però, non si registrano variazioni di rilievo rispetto alle tecniche precedenti (v. figura 7.3), il che permette di concludere ragionevolmente che il problema di distinzione tra soggetti affetti da IGT e T2D non sia risolvibile tramite tecniche di separazione lineare, almeno per quanto riguarda gli scenari esaminati.



Scenario 1	
$\varepsilon_{\mathbf{train}}$	6.5%
$\varepsilon_{\mathbf{CV}}$	27.5%
$\varepsilon_{\mathbf{test}}$	32.3%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	21	16
	T2D	4	21

Figura 7.3: regressione logistica applicata allo scenario 1.

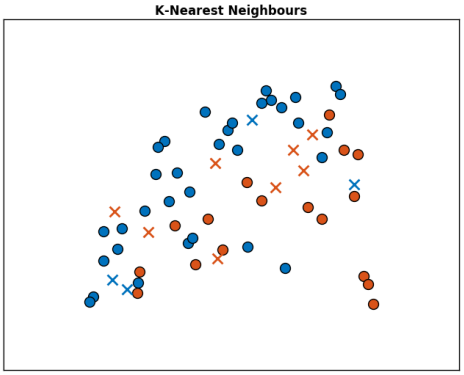
6.3 Classificazione non parametrica

I risultati ottenuti con le tecniche precedentemente utilizzate erano, probabilmente, prevedibili alla luce del fatto che nessuno degli indici di variabilità glicemica è risultato ancora preponderante rispetto agli altri nella caratterizzazione dello stato di controllo glicemico. Questa situazione esclude, infatti, una buona parte della casistica in favore della separabilità lineare che, invece, sarebbe molto più semplice da rilevare se almeno uno degli indici fosse perfettamente correlato con la classe di appartenenza dei soggetti. Si deve, perciò, ricorrere ad altri tipi di tecniche che permettano di ottenere, anziché iperpiani, superfici generiche di separazione.

6.3.1 K-Nearest Neighbours

Il metodo più semplice dal punto di vista concettuale per ottenere *boundary* non lineari tra due classi è il K-Nearest Neighbours. Si tratta di una tecnica

completamente non parametrica che si basa soltanto sull'affinità tra gli elementi del training set e i nuovi dati da classificare. Inoltre, il KNN può contare su un altissimo grado di flessibilità che gli deriva dalla possibilità di effettuare un *tuning* di due iperparametri fondamentali: il numero di vicini K e il tipo di norma p utilizzata per caratterizzare la similarità tra le osservazioni. Tutte queste caratteristiche positive, naturalmente, vengono a discapito della lunghezza della fase di test, durante la quale non si può prescindere dal confronto tra il nuovo dato e tutto il training set.
Tuttavia, guardando i risultati ci si può facilmente rendere conto del fatto che si tratti di una strategia vincente, anche limitandosi alla scansione di una griglia di 100 elementi data dall'insieme delle coppie $(K, p) \in \{1, \dots, 25\} \times \{1, 2, 4, 6\}$. Infatti, si registra un miglioramento globale e sostanziale delle performance, via via più considerevole man mano che aumenta il numero delle feature considerate.
In particolare, per quanto riguarda lo scenario 1 (indici di GV e informazioni cliniche di base), l'accuratezza ottenuta, pari a circa 81% (v. figura 7.4) fa impallidire il misero 66% della baseline per lo stesso scenario.
Tuttavia, il lettore attento e dotato di una sana dose di scetticismo avrà senz'altro notato che tale miglioramento drammatico non si riscontra anche per l'errore di cross-validazione. Ragionevolmente, però, questo fenomeno è dovuto in gran parte alla scarsa numerosità del training e del test set che vengono utilizzati per effettuare la grid search e le stime intermedie dell'errore di classificazione sulle fold. Si tratta, in fin dei conti, di un'ipotesi sensata perché il KNN è una tecnica fortemente radicata nello sfruttamento dei dati presi singolarmente, piuttosto che sintetizzati in un vettore di parametri.
Sulla scorta di quanto fin qui evidenziato, si può concludere che il problema di distinzione tra IGT e T2D sia, quantomeno, risolvibile e, anzi, si presti a una suddivisione fortemente non lineare tra le classi. Inoltre, i risultati ottenuti, in particolare, con il KNN sono confortanti circa l'utilità degli indici di variabilità glicemica per la diagnosi delle due condizioni. Infatti, se un metodo che si basa, sostanzialmente, sulla vicinanza tra i valori di variabili omologhe riesce a classificazione corretamente una gran parte dei pazienti, questo suggerisce che almeno un sottinsieme degli indici sia significativo per la caratterizzazione della patologia.



Scenario 1	
$\varepsilon_{\mathbf{train}}$	17.7%
$\varepsilon_{\mathbf{CV}}$	32.3%
$\varepsilon_{\mathbf{test}}$	19.4%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	33	4
	T2D	8	17

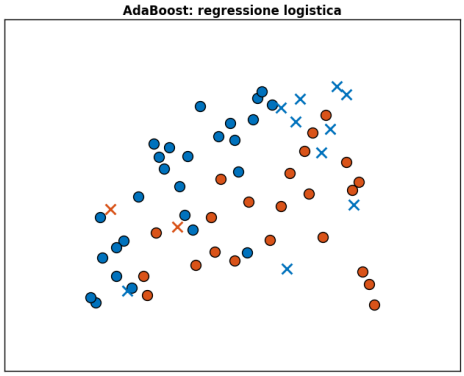
Figura 7.4: K-Nearest Neighbours applicato allo scenario 1.

6.4 Metodi di ensamble

Una strada che è possibile intraprendere per il miglioramento delle performance è quella di combinare le uscite di molti classificatori semplici ricorrendo ai metodi di ensamble. In generale, questo tipo di metodi fornisce una suddivisione del target space in regioni interpretabili come unioni di iperrettangoli e, quindi, capaci di approssimare superfici di separazione non lineari.
Nello specifico, si è ricorso a boosting e bagging applicati alla regressione logistica e agli alberi decisionali. In ambo i casi, ciascun classificatore debole ha avuto accesso a tutte le feature. Per quanto riguarda la random forest, invece, ci si è basati solo su alberi decisionali limitati a sottinsiemi di 7 feature. Inoltre, solo per quest'ultimo metodo, più promettente sulla carta, si è ricorso alla grid search per la ricerca degli iperparametri.

6.4.1 Boosting

Il boosting, come si ricorderà, è una tecnica basata sul voto di tutti i classificatori deboli pesato dalla credibilità del classificatore stesso sulla porzione di dati d'interesse. Nello specifico, i risultati riportati fanno riferimento all'algoritmo SAMME.R che, per i problemi biclasse, si riduce al semplice AdaBoost [43].
Se ci si sofferma, innanzi tutto, sul boosting della regressione logistica (v. figura 7.5) si può notare come l'accuratezza su test set sia in generale aumentata e, anzi, sia tra le migliori raggiunte fino ad ora nel caso dello scenario 2, caratterizzato dal maggior numero di variabili da esaminare. Il miglioramento delle prestazioni, purtroppo, è piuttosto variabile in entità. Si può ipotizzare che ciò sia dovuto, da un lato alla natura dell'algoritmo che, come discusso nella teoria, preclude l'interpretazione modellistica del risultato e, dall'altro, dall'iniziale inappropriatezza della regressione logistica come classificatore debole.



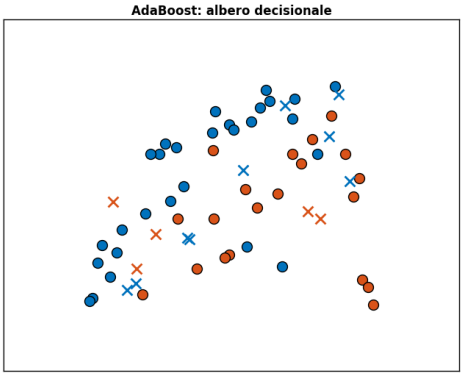
Scenario 2	
$\varepsilon_{\mathbf{train}}$	24.2%
$\varepsilon_{\mathbf{CV}}$	32.3%
$\varepsilon_{\mathbf{test}}$	19.4%

		Predizione IGT	Predizione T2D
Realtà	**IGT**	27	10
	T2D	2	23

Figura 7.5: AdaBoost (regressione logistica) applicato allo scenario 2.

La scelta dell'albero decisionale come secondo classificatore debole è motivata, semplicemente, dal desiderio di operare un confronto con la random forest che, pure, è un metodo di ensamble basato sugli alberi. Anche qui, come per il boosting della regressione logistica, il livello di accuratezza sul test set non è consistente tra i diversi scenari, probabilmente per ragioni simili. Tuttavia, si

può osservare con soddisfazione come le prestazioni nei due casi più favorevoli siano tra le migliori ottenute finora, con record provvisorio di accuratezza nel caso dello scenario 0 (v. figura 7.6).



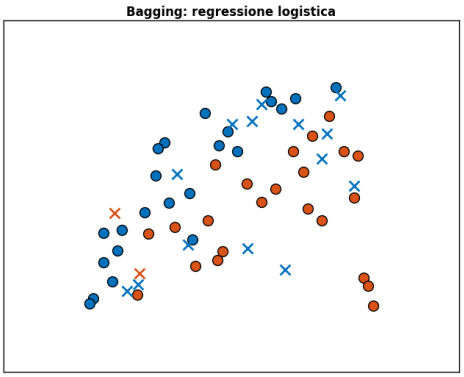
Scenario 0	
$\varepsilon_{\textbf{train}}$	0.0%
$\varepsilon_{\textbf{CV}}$	27.4%
$\varepsilon_{\textbf{test}}$	22.6%

		Predizione IGT	Predizione T2D
Realtà	**IGT**	28	9
	T2D	5	20

Figura 7.6: AdaBoost (albero decisionale) applicato allo scenario 0.

6.4.2 Bagging

Il bagging, come le altre tecniche di ensemble, è basato sulla votazione da parte di un insieme di classificatori deboli della classe di appartenenza. A differenza del boosting, però, il voto non è pesato. Inoltre, ciascun classificatore debole è applicato a un diverso sottinsieme del training set di dimensione fissa ottenuto tramite estrazione con reinserimento. Nel caso in esame, si è scelto di considerare, di volta in volta, il 50% dei dati a disposizione.
Per quanto riguarda la regressione logistica, le prestazioni sono migliori di quelle ottenute tramite AdaBoost e, senz'altro, più coerenti tra uno scenario e l'altro. In particolare, si può notare come l'aumento del numero delle variabili a disposizione porti a un aumento dell'accuratezza sul test set. Tuttavia, permane la tendenza, già incontrata in precedenza, a classificare molto bene solo una delle due classi (si veda, ad esempio, la figura 7.7).

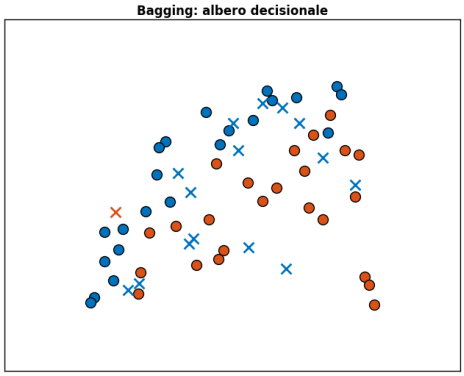


Scenario 1	
$\varepsilon_{\textbf{train}}$	14.5%
$\varepsilon_{\textbf{CV}}$	29.1%
$\varepsilon_{\textbf{test}}$	25.8%

		Predizione IGT	Predizione T2D
Realtà	**IGT**	23	14
	T2D	2	23

Figura 7.7: Bagging (regressione logistica) applicato allo scenario 1.

Per quanto riguarda il bagging di alberi decisionali, invece, le prestazioni sono in linea con quelle del boosting con lo stesso classificatore debole. Infatti, si può pensare che il leggero peggioramento nel caso degli scenari 0 e 2 venga compensato dalla maggior coerenza tra i risultati: lo scenario 1 non dà più luogo a un inaccettabile errore del 40% (v. figura 7.8).
Alla luce di quanto visto, si potrebbe, quindi, ipotizzare che il bagging sia un metodo più robusto per la soluzione del problema in esame rispetto all'applicazione di AdaBoost che, tuttavia, si dimostra in grado di dare risultati migliori nei casi più favorevoli.



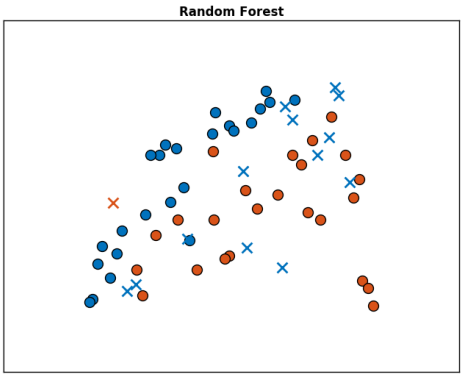
Scenario 1	
$\varepsilon_{\mathbf{train}}$	1.6%
$\varepsilon_{\mathbf{CV}}$	33.5%
$\varepsilon_{\mathbf{test}}$	25.8%

		Predizione IGT	Predizione T2D
Realtà	**IGT**	22	15
	T2D	1	24

Figura 7.8: Bagging (albero decisionale) applicato allo scenario 1.

6.4.3 Random Forest

Dal punto di vista pratico, la random forest affianca all'estrazione bootstrap degli elementi del training set, l'applicazione di ciascun classificatore debole a un sottinsieme delle feature a sua volta estratto casualmente con reinserimento. Come si è visto nel capitolo 3, il classificatore debole consigliato per questo particolare metodo di ensemble è l'albero decisionale.
Analizzando i risultati (per un esempio, si veda figura 7.9), si può, in effetti, osservare come le prestazioni siano sia buone sia coerenti. Le performance di AdaBoost nei casi favorevoli rappresentati dagli scenari 0 e 2 vengono replicate, sebbene con una maggior tendenza all'attribuzione asimmetrica tra le classi, mentre, come nel bagging, non si osservano particolari variazioni tra gli scenari. Si può ipotizzare, inoltre, che l'esito generalmente migliore della classificazione tramite random forest si possa attribuire non solo alle proprietà del metodo, ma anche all'applicazione della grid search per la ricerca degli iperparametri. In particolare, si sono fatti variare il numero di alberi utilizzati e la loro massima profondità nella griglia $\{30, \ldots, 150\} \times \{2, 3, 4\}$.



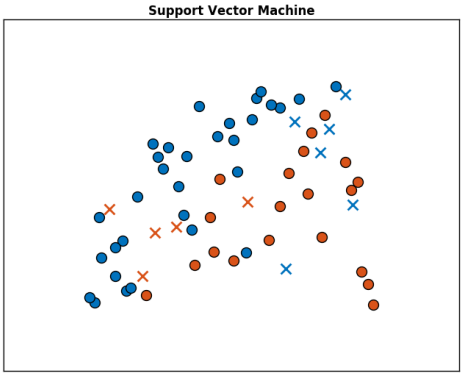
Scenario 0	
$\varepsilon_{\mathbf{train}}$	4.8%
$\varepsilon_{\mathbf{CV}}$	20.9%
$\varepsilon_{\mathbf{test}}$	22.6%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	24	13
	T2D	1	24

Figura 7.9: Random Forest applicata allo scenario 0.

6.5 Support Vector Machine

Come si è visto, il miglioramento delle performance rispetto alle tecniche più semplici può essere perseguito combinando una grande quantità di classificatori semplici attraverso i metodi di ensemble. L'alternativa è quella di utilizzare un unico classificatore, più complesso, confidando nella sua capacità di dividere correttamente il target space.
Un esempio è quello delle support vector machine che estendono, in maniera spesso proficua, il concetto di classificatore lineare parametrico.



Scenario 2	
$\varepsilon_{\mathbf{train}}$	21.0%
$\varepsilon_{\mathbf{CV}}$	32.3%
$\varepsilon_{\mathbf{test}}$	17.7%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	31	6
	T2D	5	20

Figura 7.10: Support Vector Machine applicata allo scenario 2.

6.5.1 SVM lineare

Per iniziare, si presentano i risultati dell'applicazione di una Support Vector Machine lineare, ossia di una SVM che tenta di separare le classi tramite un iperpiano.
Ci si può aspettare che i risultati siano simili a quelli ottenuti con il classificatore lineare basato sulla regressione e l'indicator matrix coding oppure tramite FLD. Proprio questo succede, infatti, negli scenari 0 e 1, laddove l'accuratezza ottenuta è comparabile. Addirittura, confrontando FLD e SVM nello scenario 1 tramite analisi delle rispettive figure riassuntive (v. appendice A), si può notare

come gli elementi del test set mal classificati siano sostanzialmente gli stessi. Tuttavia, la maggior complessità del metodo sembra dare i suoi frutti per quanto riguarda lo scenario 2, più completo, per il quale si registrano performance soddisfacenti. Infatti, oltre a una maggior accuratezza rispetto agli altri casi si può osservare un ottimo bilanciamento nel tipo di errori commessi (v. figura 7.10).
Per tutti e tre gli scenari l'unico iperparametro, il fattore di regolarizzazione C, è stato individuato tramite grid search nell'intervallo $[10^{-4}, 10^2]$.

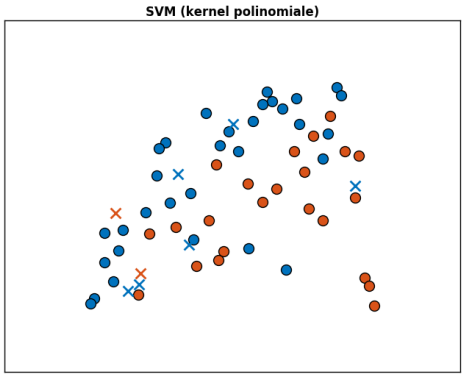
6.5.2 SVM (kernel polinomiale)

Il vero punto di forza delle support vector machine risiede nella loro flessibilità e capacità di stimare superfici di separazioni fortemente non lineari in forma parametrica grazie al kernel trick.
I risultati qui riportati fanno riferimento, in particolare, a un kernel nella forma

$$k(u, v) = (\langle u, v \rangle + c_0)^d$$

dove d e c_0, così come il termine di regolarizzazione C, sono iperparametri da individuare tramite grid search.
In fase implementativa, si sono utilizzare tre griglie differenti, scelte come compromesso tra generalità dei risultati e onere computazionale per la cross-validazione (le griglie sono tridimensionali!):

- Scenario 0: $(C, d, c_0) \in [10^{-4}, 10] \times \{3, \dots, 6\} \times [0, 10]$;
- Scenario 1: $(C, d, c_0) \in [10^{-1}, 10] \times \{3, \dots, 6\} \times [0, 10]$;
- Scenario 2: $(C, d, c_0) \in [10^{-1}, 10] \times \{3, \dots, 6\} \times [0, 10]$.



Scenario 1	
$\varepsilon_{\mathbf{train}}$	0.0%
$\varepsilon_{\mathbf{CV}}$	24.0%
$\varepsilon_{\mathbf{test}}$	12.9%

Realtà \ Predizione	IGT	T2D
IGT	31	6
T2D	2	23

Figura 7.11: Support Vector Machine (kernel polinomiale) applicata allo scenario 1.

A parte lo scenario 0, per cui l'accuratezza è ancora mediocre, si notano performance molto soddisfacenti su tutti i fronti. In particolare, la SVM con kernel polinomiale si configura come la miglior tecnica di classificazione individuata per lo scenario 1. Si tratta, anzi, della combinazione metodo-scenario caratterizzata

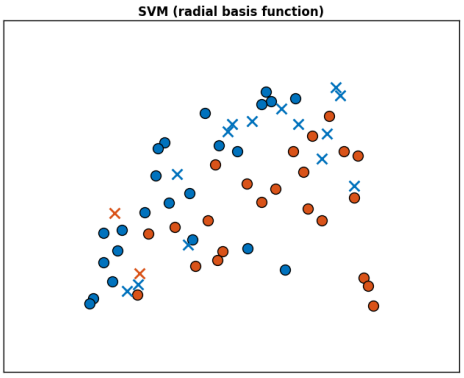
dalla massima accuratezza rispetto a tutte quelle esaminate. Infatti, con soli 8 errori sul test set, peraltro ben distribuiti tra i due tipi, esibisce prestazioni ottime, portando l'accuratezza all'87.1%, come si può osservare in figura 7.11. Leggermente peggiori i risultati per lo scenario 2, che rimangono comunque abbastanza soddiscanti, nonostante il leggero sbilanciamento tra gli errori.

6.5.3 SVM (radial basis function)

Altro kernel molto utilizzato e che, peraltro, sembra dare ottimi risultati per diversi problemi basati sulla variabilità glicemica affrontati in letteratura [24][25] [26][30] è la radial basis function, un kernel gaussiano nella forma

$$k(u, v) = e^{-\gamma \|u-v\|^2}$$

Ancora una volta, γ si aggiunge a C nel novero degli iperparametri da fissare tramite grid search (griglia per (C, γ): $[10^{-1}, 10] \times [10^{-4}, 10^{-1}]$).
Purtroppo, però, non si registra alcun miglioramento rispetto ad altre tecniche esaminate e, anzi, si può osservare, per esempio dalla figura 7.12, un'assimetra nell'attribuzione degli elementi del test set alle classi. Questo fatto può far pensare che il particolare problema in esame sia differente, almeno per quanto riguarda il kernel più appropriato, rispetto a quelli affrontati da altri.



Scenario 1	
$\varepsilon_{\mathbf{train}}$	1.6%
$\varepsilon_{\mathbf{CV}}$	25.9%
$\varepsilon_{\mathbf{test}}$	25.8%

		Predizione IGT	Predizione T2D
Realtà	**IGT**	23	14
	T2D	2	23

Figura 7.12: Support Vector Machine (radial basis function) applicata allo scenario 1.

6.6 Riepilogo dei risultati

In tabella 6.1 è riportato ε_{test}, l'errore di classificazione sul test set, per ciascuna combinazione classificatore-scenario. Si ottiene così una visione d'insieme dei risultati che facilita l'individuazione di pattern inter- e intra-scenario.
Innanzi tutto, riassumendo alcune considerazioni fatte in precedenza, si può notare come i classificatori lineari più semplici esibiscano performance insoddisfacenti in tutti i casi, con la probabilmente fortuita eccezione di FLD per lo scenario 2 che, si ricorda, però, dà luogo a errori sbilanciati tra le classi. Inoltre, il numero di feature considerate non sembra influire in alcun modo sull'accuratezza, il che porta a pensare che siano proprio le tecniche ad essere inadatte al problema.

Classificatore	Scenario 0	Scenario 1	Scenario 2
Classificatore lineare	38.7%	33.9%	37.1%
Fisher's Linear Discriminant	30.6%	33.9%	25.8%
Regressione logistica	35.5%	32.3%	33.9%
K-Nearest Neighbours	25.8%	19.4%	17.7%
Boosting (regr. logistica)	27.4%	29.0%	19.4%
Boosting (albero)	22.6%	40.3%	21.0%
Bagging (regr. logistica)	27.4%	25.8%	21.0%
Bagging (albero)	24.2%	25.8%	24.2%
Random Forest	22.6%	27.4%	21.0%
Support Vector Machine	32.3%	35.5%	17.7%
SVM (kernel polinomiale)	25.8%	12.9%	17.7%
SVM (radial basis function)	24.2%	25.8%	22.6%

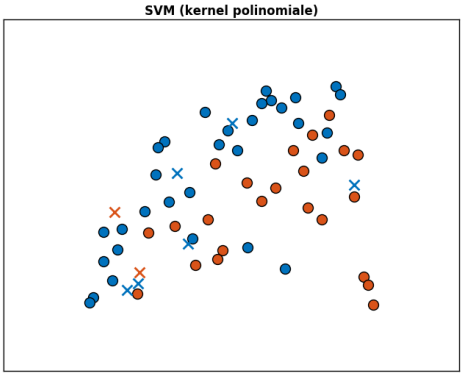
Tabella 6.1: Test error per il problema di classificazione IGT vs T2D sugli scenari 0, 1 e 2.

Diversamente, il KNN esibisce un comportamento piuttosto favorevole dal punto di vista dei risultati, oltre che una diminuzione dell'errore al crescere della complessità dello scenario. Come discusso in precedenza, si può ipotizzare che ciò sia dovuto a una ragionevole similarità di comportamento all'interno di ciascuna delle classi dei parametri considerati. Si ricorda, però, che il KNN è caratterizzato da scarsa scalabilità: a differenza dei metodi parametrici che, a valle della fase di training, possono fornire rapidamente una classificazione di dati ignoti, il KNN paga la rapidità del training (che consiste, semplicemente nella memorizzazione di tutto il set) con tempi potenzialmente molto lunghi di test, dovuti alla inevitabile necessità di confrontare la nuova osservazione con tutte le precedenti. Dunque, in un'ottica di grandi numeri, potrebbe risultare ingestibile in termini sia di complessità, sia di portabilità (per condividere il classificatore, si deve trasferire l'intero training set).
Rispetto ai classificatori lineari che costituiscono la baseline di performance, i metodi di ensemble danno risultati generalmente migliori, come, peraltro, è lecito aspettarsi. In particolare, il boosting, pur detenendo il record di accuratezza nella sua categoria quando è applicato allo scenario 2, non sembra eccellere per robustezza, viste le prestazioni altalenanti. Al contrario, sia il bagging, sia la random forest portano a risultati molto più coerenti tra loro. Dunque, in generale, i metodi di ensemble paiono essere più favorevoli allo scenario 2, probabilmente perché in grado di sfruttare al meglio l'informazione portata da un maggior numero di feature.
Osservando, infine, i risultati dell'applicazione delle support vector machine si può notare come la maggior complessità del classificatore porti, in effetti, alcuni vantaggi. In particolare, la SVM con kernel polinomiale riferita allo scenario 1 detiene il primato di accuratezza rispetto alle 36 situazione esaminate. Si tratta, peraltro, di un risultato soddisfacente in senso assoluto poiché prossimo al 90% nonostante l'esigua dimensione del training set.
Il fatto che, poi, la combinazione di variabili vincente sia proprio quella dello scenario 1 è particolarmente positivo. Sebbene, infatti, le prestazioni generali sullo scenario 0 siano mediocri, esse possono essere radicalmente migliorate ag-

giungendo ai soli indici di variabilità glicemica estratti dal segnale CGM alcune informazioni sul paziente, ottenibili a costo zero.
Tuttavia, è il caso di evidenziare anche il fatto che, nonostante l'accuratezza raggiunta non sia eccelsa, lo scenario 2 esibisce la massima coerenza dei risultati, più che buoni *across the board*. Si può, quindi, ipotizzare che, volendo generalizzare i risultati ottenuti estendendoli a una popolazione più ampia, si dovrà raggiungere un compromesso tra conforto statistico circa la robustezza dei dati e costo di estrazione dei parametri clinici aggiuntivi che distinguono lo scenario 2 dallo scenario 1.
Infine, emerge il fatto che i soli indici di variabilità glicemica, almeno in un contesto *data poor*, siano inadeguati a differenziare correttamente tra IGT e T2D. Fortunatamente, però, è sufficiente complementare gli indici di GV con alcune informazioni cliniche di base per aumentare vertiginosamente le performance.

6.7 Risultati con feature selection

Tutti gli scenari visti finora hanno la caratteristica comune di utilizzare tutti gli indici di variabilità glicemica presentati nel capitolo 2, opportunamente corredati degli appropriati parametri clinici. Tuttavia, ha senso chiedersi se tutte le variabili in gioco siano effettivamente significative per risolvere il problema di classificazione. Per rispondere a questo interrogativo, come accennato nella sezione 4.5, si può ricorrere a metodi automatici o manuali.
Per non rischiare di ripetere considerazioni spesso molto simili, nel seguito si riporteranno solo i risultati salienti e, dunque, meritevoli di commento.



Scenario 1	
$\varepsilon_{\mathbf{train}}$	1.6%
$\varepsilon_{\mathbf{CV}}$	29.0%
$\varepsilon_{\mathbf{test}}$	32.3%

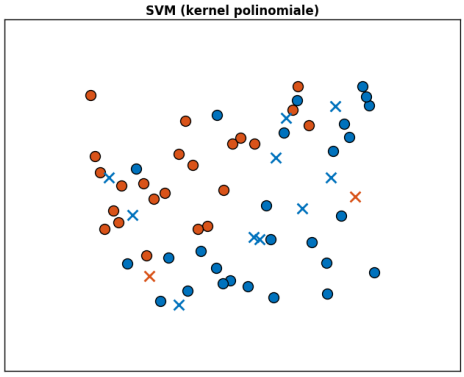
		Predizione	
		IGT	**T2D**
Realtà	**IGT**	20	17
	T2D	3	22

Figura 7.13: Support Vector Machine (kernel polinomiale), sottoposta a Recursive Feature Elimination e applicata allo scenario 1.

6.7.1 Feature selection automatica

In generale, i risultati della feature selection automatica, sia realizzata tramite recursive feature elimination sia incorporata nella funzione costo tramite penalizzazione L1 non sono per nulla soddisfacenti.
A titolo di esempio, in figura 7.13 si presentano le performance per una SVM con kernel polinomiale sottoposta a RFE per individuare le 25 feature più importanti tra le 41 dello scenario 1. In tutti gli altri casi, non solo il livello di accuratezza raggiunto è molto simile, ma, addirittura, si presta ai medesimi

commenti. In particolare, si può notare che le prestazioni, nonostante ancora accettabili, sono peggiorate notevolmente rispetto a quelle ottenute con tutte le feature dello scenario. Inoltre, si assiste a un ritorno dello sbilanciamento tra i tipi di errore commesso.
Risultati di questo tipo, non molto buoni, sono probabilmente ascrivibili all'eccessiva riduzione della porzione di dati adibiti a training set. Infatti, rispetto a quanto descritto nella sezione 5.6, è necessario innestare un'ulteriore cross-validazione all'interno del processo di calcolo per permettere una corretta identificazione delle feature più significative.



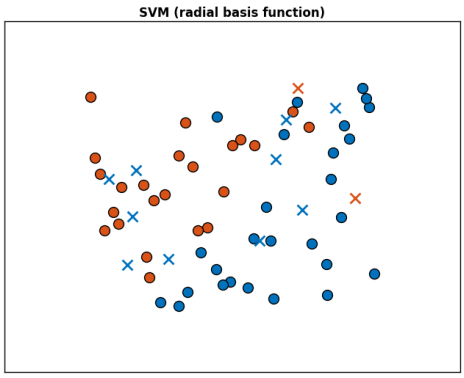
Scenario 3	
$\varepsilon_{\mathbf{train}}$	1.6%
$\varepsilon_{\mathbf{CV}}$	25.8%
$\varepsilon_{\mathbf{test}}$	19.4%

Realtà \ Predizione	IGT	T2D
IGT	27	10
T2D	2	23

Figura 7.14: Support Vector Machine (kernel polinomiale) applicata allo scenario 3.

6.7.2 Expert knowledge

Si è reso necessario, dunque, ricorrere all'introduzione nel processo di classificazione dell'expert knowledge fornita da uno specialista diabetologo come descritto nella sezione 5.4. Di seguito si fa riferimento all'applicazione proprio sullo scenario 3 dei tre tipi di SVM considerati.



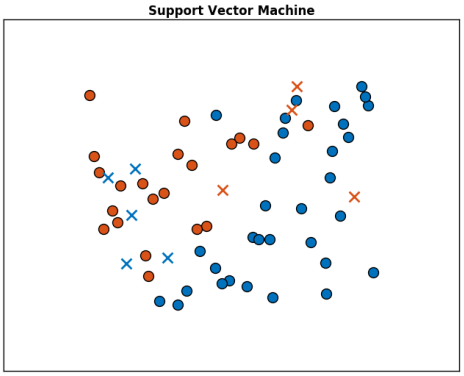
Scenario 3	
$\varepsilon_{\mathbf{train}}$	9.7%
$\varepsilon_{\mathbf{CV}}$	25.5%
$\varepsilon_{\mathbf{test}}$	19.4%

Realtà \ Predizione	IGT	T2D
IGT	27	10
T2D	2	23

Figura 7.15: Support Vector Machine (radial basis function) applicata allo scenario 3.

Procedendo in ordine discendente rispetto all'accuratezza precedente, in figura 7.14 si può notare come la SVM con kernel polinomiale abbia perso la sua posizione preminente: nonostante risultati ancora buoni, sebbene peggiori, viene

meno la distribuzione equilibrata degli errori.
Per quanto riguarda il kernel rbf si registra, invece, un leggero miglioramento, peraltro di scarso interesse perché permane l'asimmetria nell'attribuzione alle classi (v. figura 7.15).
A sorpresa, però, la semplice SVM lineare esibisce un comportamento davvero favorevole. Infatti, con un solo errore in più rispetto alla SVM con kernel polinomiale applicata allo scenario 1, raggiunge ottimi livelli di accuratezza. Inoltre, nonostante non sembri, a prima vista, un risultato particolarmente degno di nota, questo fatto ha delle ripercussioni fondamentali per quanto riguarda l'interpretabilità dei dati. Infatti, come si è detto nel capitolo 3, i classificatori lineari (quali la SVM semplice) si prestano a interpretazioni modellistiche e probabilistiche che, al contrario, sono impossibili da fornire una volta applicato il kernel trick. Purtroppo, però, il numero limitato di dati a disposizione impedisce di trarre conclusioni definitive in proposito.
Tuttavia, rimane fermo il fatto che, con un'opportuna selezione delle variabili, si può ricondurre un problema apparentemente irrisolvibile con tecniche lineari alla semplice ricerca di un iperpiano separatore, senza dover scomodare metodi più complessi.



Scenario 3	
$\varepsilon_{\mathbf{train}}$	19.4%
$\varepsilon_{\mathbf{CV}}$	28.9%
$\varepsilon_{\mathbf{test}}$	14.5%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	32	5
	T2D	4	21

Figura 7.16: Support Vector Machine applicata allo scenario 3.

In coda, si riportano, per completezza, gli estremi delle grid search effettuate.

- SVM: $C \in [10^{-4}, 10]$;
- SVM, kernel polinomiale: $(C, d, c_0) \in [10^{-4}, 10] \times \{10^{-4}, 10\} \times [0, 10]$;
- SVM, radial basis function: $(C, \gamma) \in [10^{-1}, 10] \times [10^{-4}, 10^{-1}]$).

Capitolo 7

Conclusioni

Lo sviluppo delle tecnologie per il monitoraggio in continua del glucosio sta spingendo verso l'adozione di nuove metriche per caratterizzare il livello di controllo della glicemia. In questo contesto si inserisce il concetto di GV che sembra destinato a diventare un indicatore fondamentale, in virtù della sua attitudine a complementare le informazioni fornite dall'emoglobina glicata. Come si è visto, però, a tutt'oggi manca un consenso definitivo su quali (combinazioni di) indici di GV siano i più affidabili e, addirittura, sul modo in cui essi debbano essere calcolati.
L'obiettivo principale di questo lavoro è stato quello di dimostrare, almeno a livello di *proof of concept*, quali possano essere le prestazioni di un sistema diagnostico basato proprio sulla variabilità glicemica. A questo scopo si è affrontato il problema di classificare pazienti affetti da IGT e T2D nelle rispettive categorie tramite tecniche adatte anche in caso di scarsa numerosità dei dati.
Si sono predisposti, inoltre, tre scenari di classificazione, corrispondenti a diversi gradi di conoscenza circa lo stato di salute del paziente di cui si suppone di avere a disposizione, inizialmente, solo il tracciato CGM da cui calcolare gli indici di GV, poi anche una ridotta selezione di parametri clinici di base e, infine, gli esiti di alcuni semplici esami diagnostici. Per ciascuno degli scenari è stata applicata un'ampia collezione di classificatori (tragli altri: regressione logistica, KNN, metodi di ensemble e SVM), scelti in modo progressivo sulla base di ragionevoli ipotesi sull'inefficacia o inadeguatezza delle soluzioni precedenti.
I risultati dell'analisi sono stati senz'altro soddisfacenti. Infatti, sebbene i soli indici di variabilità glicemica non siano particolarmente indicati per la classificazione in un contesto *data poor*, è sufficiente arricchirli di alcuni parametri clinici di base per migliorare notevolmente la situazione. Nello specifico, l'aggiunta di età, sesso, indice di massa corporea e circonferenza della vita permette di passare da un'accuratezza in generali pari al 75% ad appena il 12.9% di errore. Tale risultato è stato possibile grazie a una SVM con kernel polinomiale, sottoposta grid search per la stima degli iperparametri.
Si è, poi, individuato che un sottinsieme degli indici di variabilità glicemica, ancora una volta corredati dai quattro parametri appena citati oltre che dal livello di glucosio a digiuno, permette di riformulare il problema in termini di separazione lineare tra le classi. Infatti, si è ottenuta un'accuratezza dell'85.5% tramite una semplice SVM lineare.
Si tratta di risultati incoraggianti da tutti i punti di vista. In primo luogo,

infatti, a differenza di alcuni tentativi effettuati in letteratura, non sussistono ambiguità circa la classificazione dei soggetti, poiché si è tentato di replicare il risultato di procedimenti diagnostici accettati come gold standard. Proprio per questo, dunque, è difficile dubitare della capacità descrittiva degli indici di variabilità glicemica, almeno limitatamente allo stato di avanzamento del T2D. Inoltre, si è introdotta la possibilità di poter fornire interpretazioni modellistiche e probabilistiche dei risultati ottenuti, in virtù dell'impiego con successo di una tecnica di separazione lineare.

7.1 Sviluppi futuri

In prosecuzione di quanto presentato in questo lavoro, si potrebbe estendere l'applicazione degli indici di variabilità glicemica a una popolazione molto più numerosa. In questo modo, si potrebbe individuare più facilmente la migliore combinazione di indici per la valutazione dello stato di avanzamento del diabete.
Un obiettivo da perseguire potrebbe essere quello di utilizzare la GV per affrontare un problema non di classificazione, ma di regressione, tale per cui si possa seguire il miglioramento o peggioramento delle condizioni di salute di un soggetto in maniera molto più fine che tramite il solo utilizzo dell'emoglobina glicata. Idealmente, quindi, il paziente, dotato di un sensore CGM, potrebbe conoscere, con cadenza settimanale, quale sia il suo scostamento dallo stato di perfetto controllo glicemico e, conseguentemente, valutare con quale frequenza sottoporsi ad analisi più approfondite.
A questo punto, si aprirebbe l'ulteriore problema di definire in maniera oggettiva una scala di gravità del diabete con criteri simili a quelli che si utilizzano per la distinzione tra IGT e T2D tramite gli attuali gold standard.

Appendici

Appendice A

Risultati

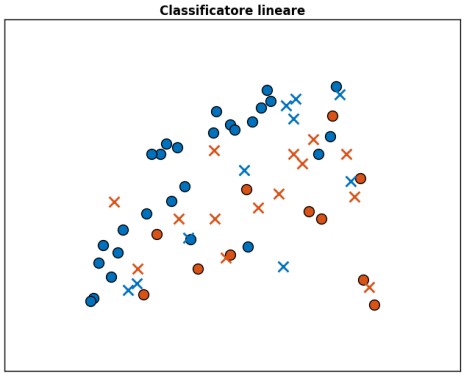
In questa appendice si riportano per intero le tavole riassuntive dei risultati ottenuti. Si ricorda (v. capitolo 6) che per ogni scenario e ogni classificatore sono a disposizione:

- una rappresentazione pittorica nello spazio bidimensionale dei risultati della classificazione sul test set;
- una tabella con l'errore percentuale di training, test e cross-validazione;
- la matrice di confusione relativa al test set.

Per interpretare l'immagine, si tenga conto che essa è stata ottenuta proiettando tramite *Principal Component Analysis* [35] i dati in due dimensioni e, successivamente, equalizzando l'istogramma delle coordinate dei punti trasformati [42]. Va da sé che la rappresentazione è poco significativa dal punto di vista matematico, ma, nonostante ciò, può essere utile per avere un riscontro visivo su quali siano, nello specifico, le osservazioni correttamente classificate.
I simboli utilizzati nell'immagine sono, intuitivamente:

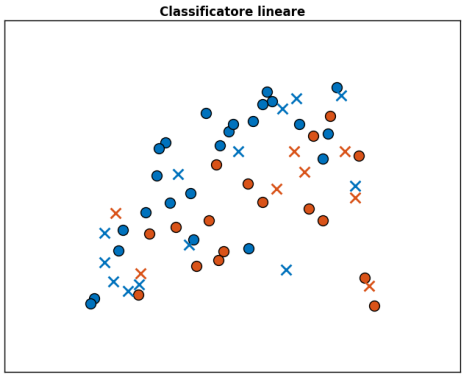
- pallino pieno in caso di classificazione corretta;
- croce in presenza di un errore di classificazione;
- colore blu (`#0072BD`) se IGT è la classe reale di appartenenza;
- colore arancio (`#D95319`) se T2D è la classe reale di appartenenza.

A.1 Classificatore lineare



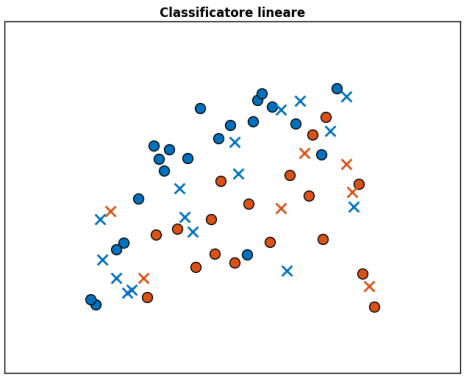
Scenario 0	
$\varepsilon_{\text{train}}$	9.7%
ε_{CV}	32.4%
$\varepsilon_{\text{test}}$	38.7%

Realtà \ Predizione	IGT	T2D
IGT	27	10
T2D	14	11



Scenario 1	
$\varepsilon_{\text{train}}$	3.2%
ε_{CV}	24.4%
$\varepsilon_{\text{test}}$	33.9%

Realtà \ Predizione	IGT	T2D
IGT	24	13
T2D	8	17

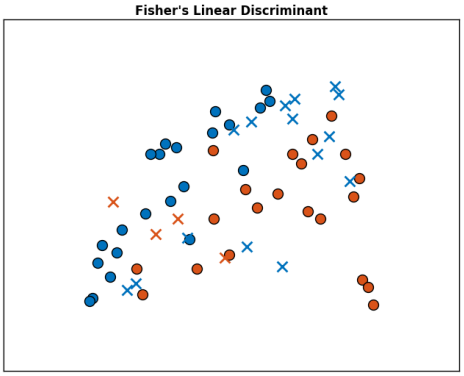


Scenario 2	
$\varepsilon_{\text{train}}$	3.2%
ε_{CV}	42.1%
$\varepsilon_{\text{test}}$	37.1%

Realtà \ Predizione	IGT	T2D
IGT	21	16
T2D	7	18

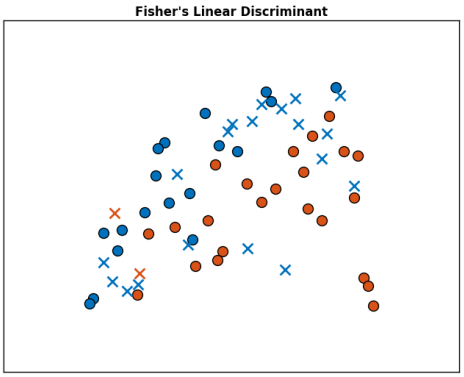
Tavola 1: Classificatore lineare con indicator matrix coding applicato agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.2 Fisher's Linear Discriminant



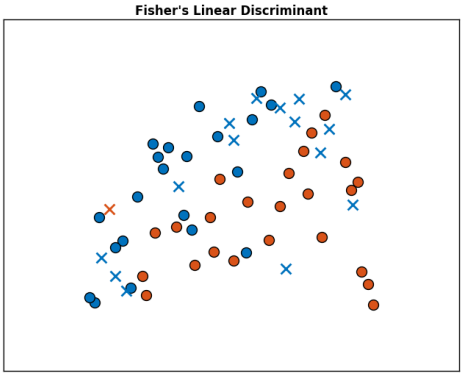
Scenario 0	
$\varepsilon_{\text{train}}$	21.0%
ε_{CV}	28.9%
$\varepsilon_{\text{test}}$	30.6%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	22	15
	T2D	4	21



Scenario 1	
$\varepsilon_{\text{train}}$	14.5%
ε_{CV}	30.7%
$\varepsilon_{\text{test}}$	33.9%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	18	19
	T2D	2	23



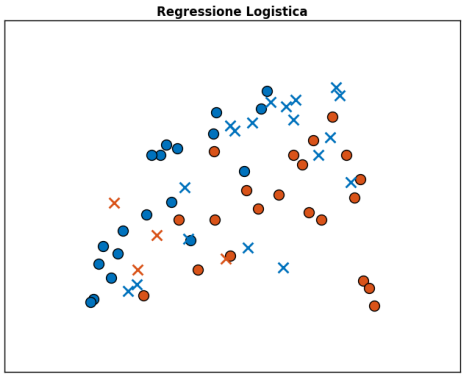
Scenario 2	
$\varepsilon_{\text{train}}$	11.3%
ε_{CV}	27.6%
$\varepsilon_{\text{test}}$	25.8%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	22	15
	T2D	1	24

Scenario 2: Fisher's Linear Discriminant.

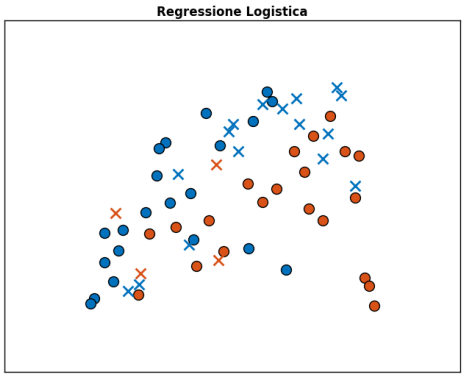
Tavola 2: Fisher's Linear Discriminant applicato agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.3 Regressione logistica



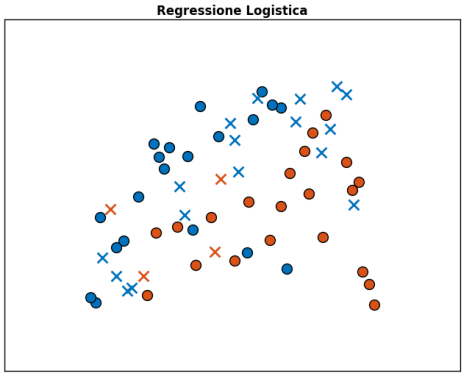
Scenario 0	
$\varepsilon_{\text{train}}$	11.3%
ε_{CV}	28.9%
$\varepsilon_{\text{test}}$	35.5%

Realtà \ Predizione	IGT	T2D
IGT	19	18
T2D	4	21



Scenario 1	
$\varepsilon_{\text{train}}$	6.5%
ε_{CV}	27.5%
$\varepsilon_{\text{test}}$	32.3%

Realtà \ Predizione	IGT	T2D
IGT	21	16
T2D	4	21

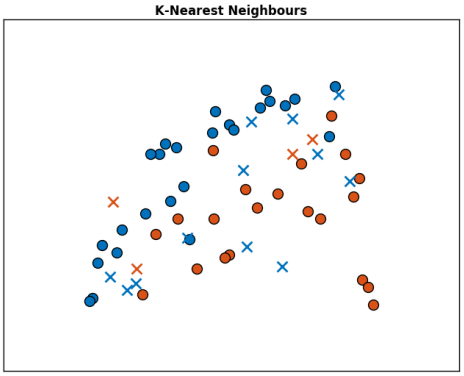


Scenario 2	
$\varepsilon_{\text{train}}$	0.0%
ε_{CV}	32.3%
$\varepsilon_{\text{test}}$	33.9%

Realtà \ Predizione	IGT	T2D
IGT	20	17
T2D	4	21

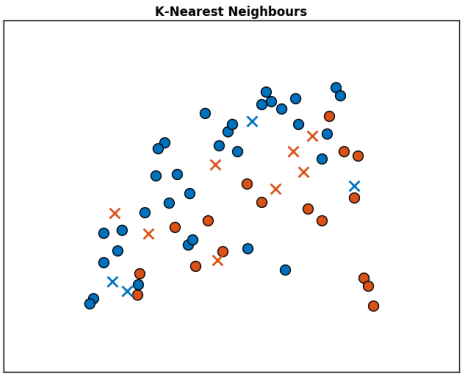
Tavola 3: Regressione logistica applicata agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.4 K-Nearest Neighbours



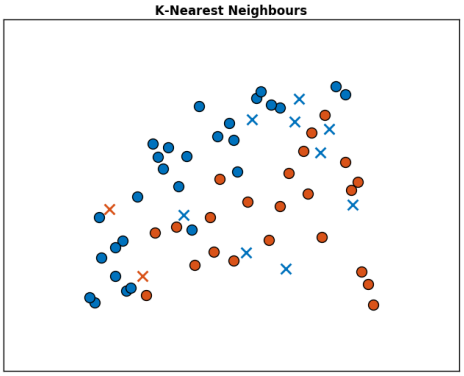
Scenario 0	
$\varepsilon_{\textbf{train}}$	17.7%
$\varepsilon_{\textbf{CV}}$	37.1%
$\varepsilon_{\textbf{test}}$	25.8%

Realtà \ Predizione	IGT	T2D
IGT	25	12
T2D	4	21



Scenario 1	
$\varepsilon_{\textbf{train}}$	17.7%
$\varepsilon_{\textbf{CV}}$	32.3%
$\varepsilon_{\textbf{test}}$	19.4%

Realtà \ Predizione	IGT	T2D
IGT	33	4
T2D	8	17

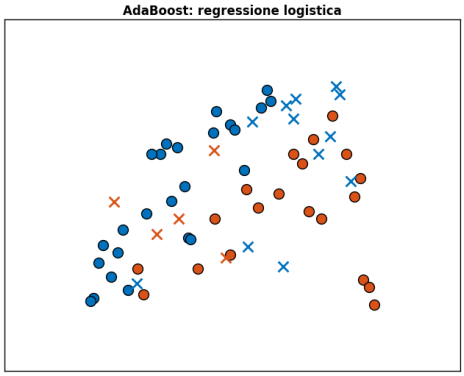


Scenario 2	
$\varepsilon_{\textbf{train}}$	21.0%
$\varepsilon_{\textbf{CV}}$	27.4%
$\varepsilon_{\textbf{test}}$	17.7%

Realtà \ Predizione	IGT	T2D
IGT	28	9
T2D	2	23

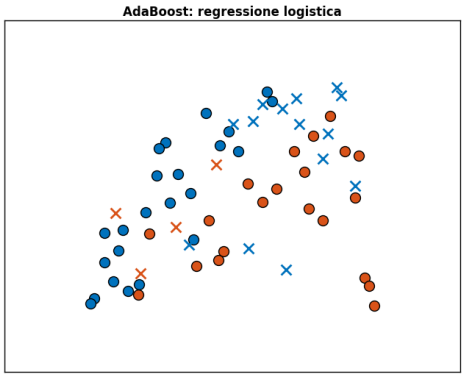
Tavola 4: K-Nearest Neighbours applicato agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.5 AdaBoost (regressione logistica)



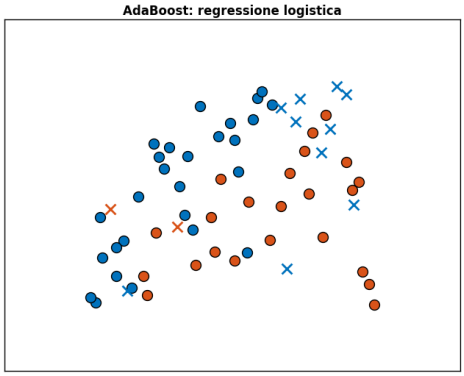
Scenario 0	
$\varepsilon_{\text{train}}$	29.0%
ε_{CV}	33.8%
$\varepsilon_{\text{test}}$	27.4%

Realtà \ Predizione	IGT	T2D
IGT	25	12
T2D	5	20



Scenario 1	
$\varepsilon_{\text{train}}$	27.4%
ε_{CV}	34.0%
$\varepsilon_{\text{test}}$	29.0%

Realtà \ Predizione	IGT	T2D
IGT	23	14
T2D	4	21

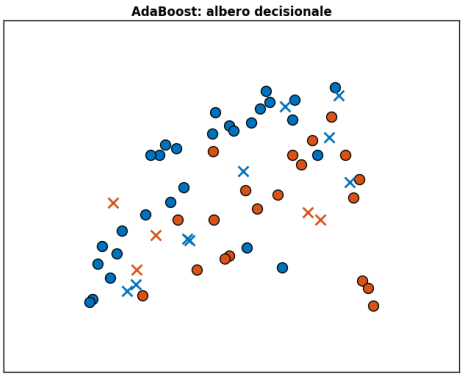


Scenario 2	
$\varepsilon_{\text{train}}$	24.2%
ε_{CV}	32.3%
$\varepsilon_{\text{test}}$	19.4%

Realtà \ Predizione	IGT	T2D
IGT	27	10
T2D	2	23

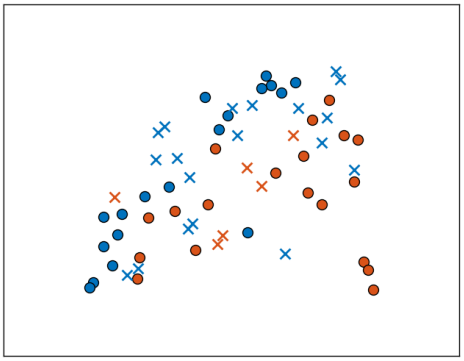
Tavola 5: AdaBoost (regressione logistica) applicato agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.6 AdaBoost (albero decisionale)



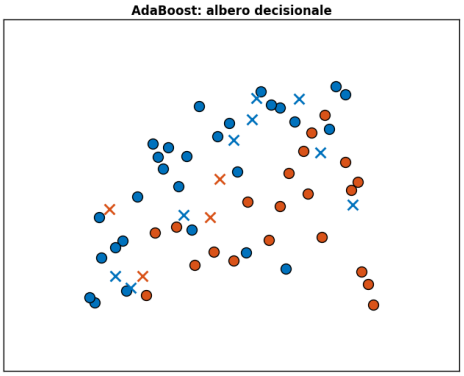
Scenario 0	
$\varepsilon_{\mathbf{train}}$	0.0%
$\varepsilon_{\mathbf{CV}}$	27.4%
$\varepsilon_{\mathbf{test}}$	22.6%

Realtà \ Predizione	IGT	T2D
IGT	28	9
T2D	5	20



Scenario 1	
$\varepsilon_{\mathbf{train}}$	0.0%
$\varepsilon_{\mathbf{CV}}$	38.8%
$\varepsilon_{\mathbf{test}}$	40.3%

Realtà \ Predizione	IGT	T2D
IGT	18	19
T2D	6	19

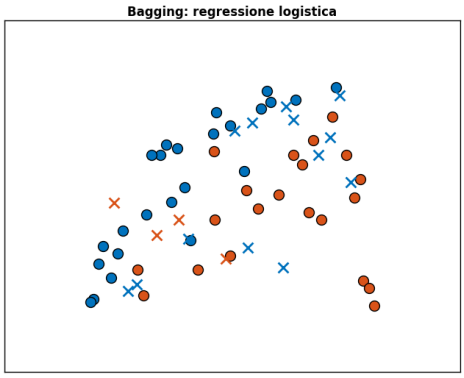


Scenario 2	
$\varepsilon_{\mathbf{train}}$	0.0%
$\varepsilon_{\mathbf{CV}}$	27.1%
$\varepsilon_{\mathbf{test}}$	21.0%

Realtà \ Predizione	IGT	T2D
IGT	28	9
T2D	4	21

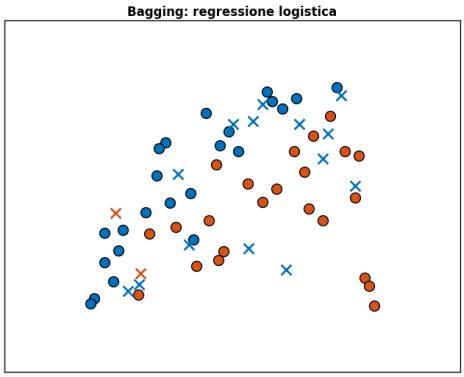
Tavola 6: AdaBoost (albero decisionale) applicato agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.7 Bagging (regressione logistica)



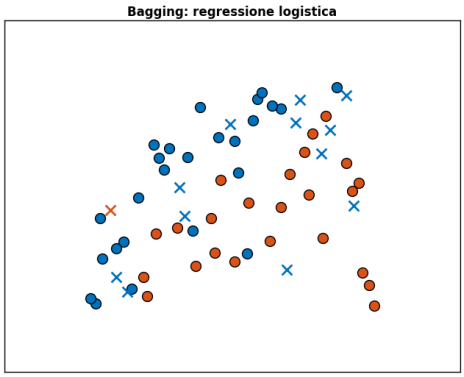
Scenario 0	
$\varepsilon_{\text{train}}$	19.4%
ε_{CV}	28.7%
$\varepsilon_{\text{test}}$	27.4%

Realtà \ Predizione	IGT	T2D
IGT	24	13
T2D	4	21



Scenario 1	
$\varepsilon_{\text{train}}$	14.5%
ε_{CV}	29.1%
$\varepsilon_{\text{test}}$	25.8%

Realtà \ Predizione	IGT	T2D
IGT	23	14
T2D	2	23

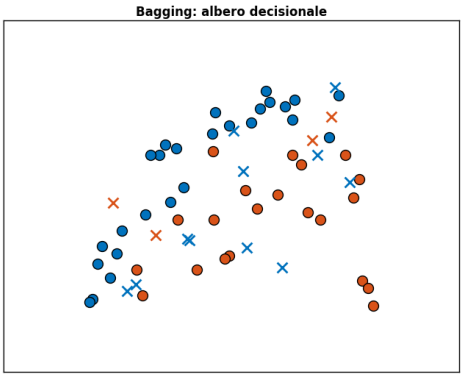


Scenario 2	
$\varepsilon_{\text{train}}$	12.9%
ε_{CV}	27.5%
$\varepsilon_{\text{test}}$	21.0%

Realtà \ Predizione	IGT	T2D
IGT	25	12
T2D	1	24

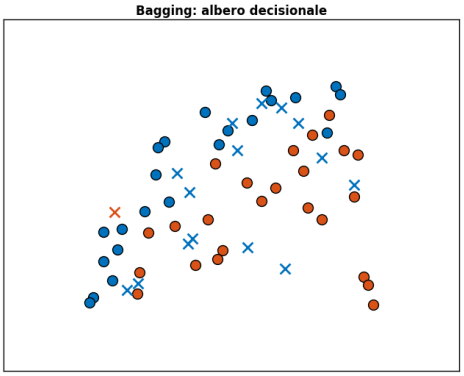
Tavola 7: Bagging (regressione logistica) applicato agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.8 Bagging (albero decisionale)



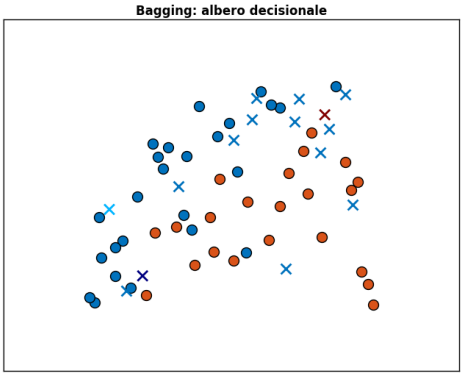
Scenario 0	
$\varepsilon_{\text{train}}$	3.2%
ε_{CV}	30.4%
$\varepsilon_{\text{test}}$	24.2%

Realtà \ Predizione	IGT	T2D
IGT	26	11
T2D	4	21



Scenario 1	
$\varepsilon_{\text{train}}$	1.6%
ε_{CV}	33.5%
$\varepsilon_{\text{test}}$	25.8%

Realtà \ Predizione	IGT	T2D
IGT	22	15
T2D	1	24

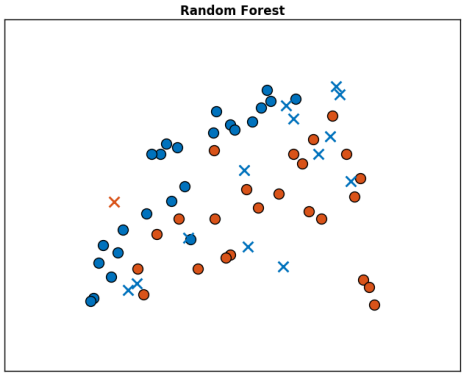


Scenario 2	
$\varepsilon_{\text{train}}$	1.6%
ε_{CV}	28.9%
$\varepsilon_{\text{test}}$	24.2%

Realtà \ Predizione	IGT	T2D
IGT	25	12
T2D	3	22

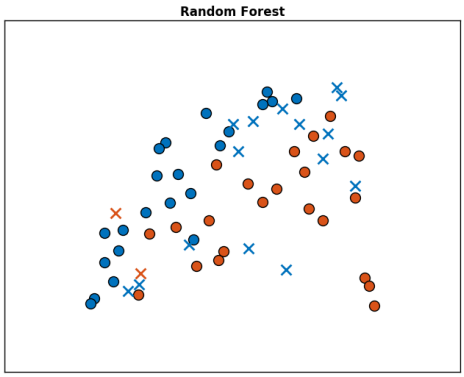
Tavola 8: Bagging (albero decisionale) applicato agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.9 Random Forest



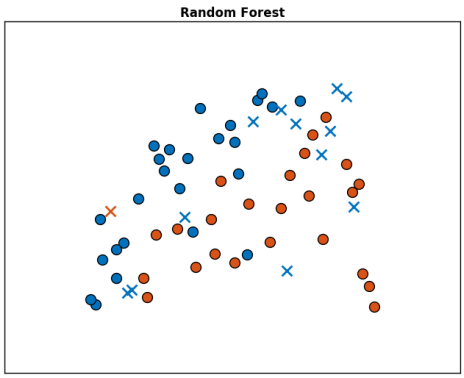
Scenario 0	
$\varepsilon_{\mathbf{train}}$	4.8%
$\varepsilon_{\mathbf{CV}}$	20.9%
$\varepsilon_{\mathbf{test}}$	22.6%

Realtà \ Predizione	IGT	T2D
IGT	24	13
T2D	1	24



Scenario 1	
$\varepsilon_{\mathbf{train}}$	0.0%
$\varepsilon_{\mathbf{CV}}$	28.9%
$\varepsilon_{\mathbf{test}}$	27.4%

Realtà \ Predizione	IGT	T2D
IGT	22	15
T2D	2	23

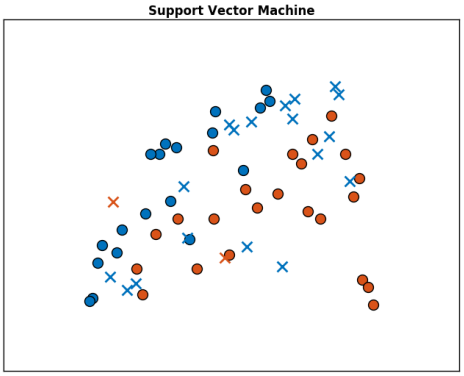


Scenario 2	
$\varepsilon_{\mathbf{train}}$	1.6%
$\varepsilon_{\mathbf{CV}}$	30.3%
$\varepsilon_{\mathbf{test}}$	21.0%

Realtà \ Predizione	IGT	T2D
IGT	25	12
T2D	1	24

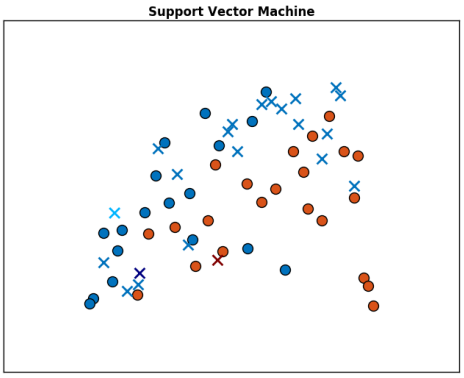
Tavola 9: Random Forest applicata agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.10 Support Vector Machine



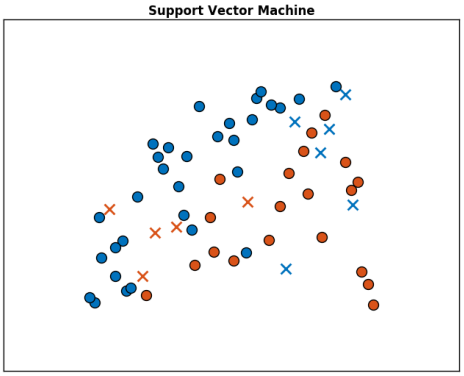
Scenario 0	
$\varepsilon_{\mathbf{train}}$	12.9%
$\varepsilon_{\mathbf{CV}}$	30.3%
$\varepsilon_{\mathbf{test}}$	32.3%

Realtà \ Predizione	IGT	T2D
IGT	19	18
T2D	2	23



Scenario 1	
$\varepsilon_{\mathbf{train}}$	6.5%
$\varepsilon_{\mathbf{CV}}$	29.0%
$\varepsilon_{\mathbf{test}}$	35.5%

Realtà \ Predizione	IGT	T2D
IGT	18	19
T2D	3	22

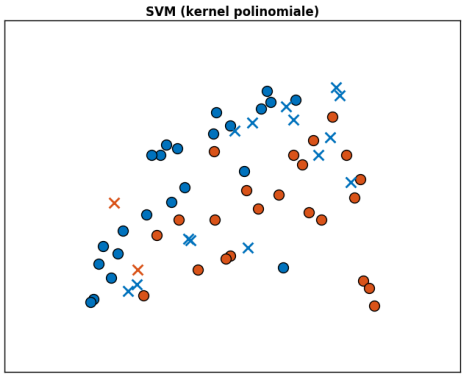


Scenario 2	
$\varepsilon_{\mathbf{train}}$	21.0%
$\varepsilon_{\mathbf{CV}}$	32.3%
$\varepsilon_{\mathbf{test}}$	17.7%

Realtà \ Predizione	IGT	T2D
IGT	31	6
T2D	5	20

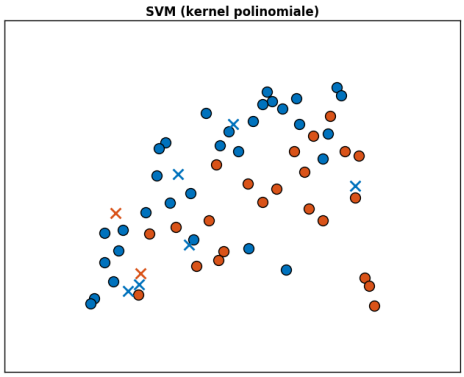
Tavola 10: Support Vector Machine applicata agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.11 SVM (kernel polinomiale)



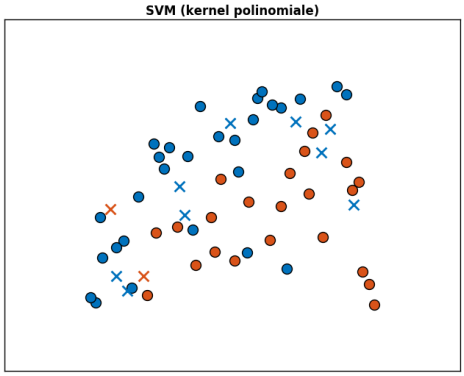
Scenario 0	
$\varepsilon_{\text{train}}$	6.5%
ε_{CV}	28.9%
$\varepsilon_{\text{test}}$	25.8%

Realtà \ Predizione	IGT	T2D
IGT	23	14
T2D	2	23



Scenario 1	
$\varepsilon_{\text{train}}$	0.0%
ε_{CV}	24.0%
$\varepsilon_{\text{test}}$	12.9%

Realtà \ Predizione	IGT	T2D
IGT	31	6
T2D	2	23

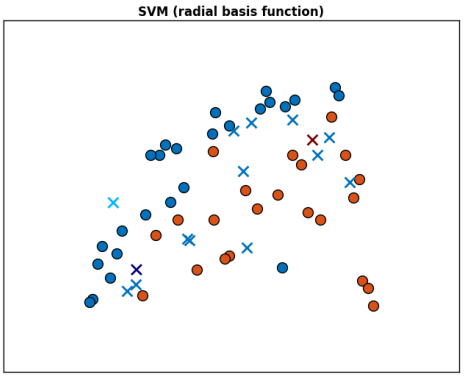


Scenario 2	
$\varepsilon_{\text{train}}$	3.2%
ε_{CV}	27.6%
$\varepsilon_{\text{test}}$	17.7%

Realtà \ Predizione	IGT	T2D
IGT	28	9
T2D	2	23

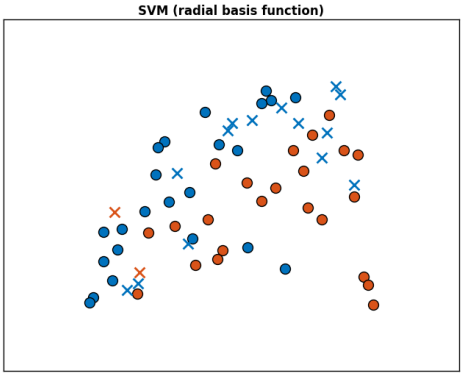
Tavola 11: Support Vector Machine (kernel polinomiale) applicata agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.12 SVM (radial basis function)



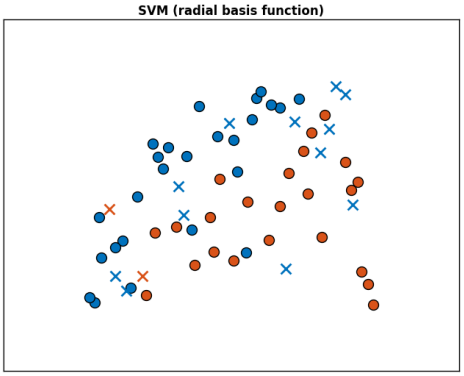
Scenario 0	
$\varepsilon_{\text{train}}$	4.8%
ε_{CV}	20.9%
$\varepsilon_{\text{test}}$	24.2%

Realtà \ Predizione	IGT	T2D
IGT	25	12
T2D	3	22



Scenario 1	
$\varepsilon_{\text{train}}$	1.6%
ε_{CV}	25.9%
$\varepsilon_{\text{test}}$	25.8%

Realtà \ Predizione	IGT	T2D
IGT	23	14
T2D	2	23

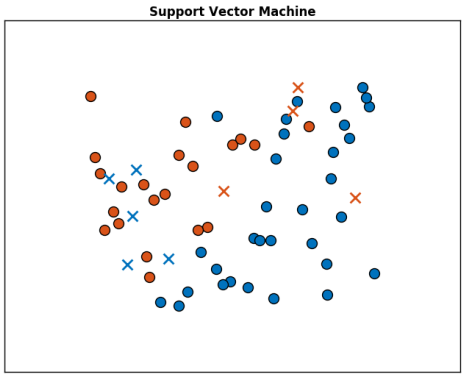


Scenario 2	
$\varepsilon_{\text{train}}$	3.2%
ε_{CV}	30.7%
$\varepsilon_{\text{test}}$	22.6%

Realtà \ Predizione	IGT	T2D
IGT	25	12
T2D	2	23

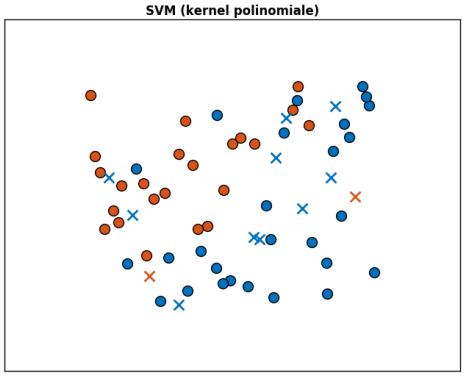
Tavola 12: Support Vector Machine (radial basis function) applicata agli scenari 0 (in alto), 1 (al centro) e 2 (in basso).

A.13 Feature selection (expert knowledge)



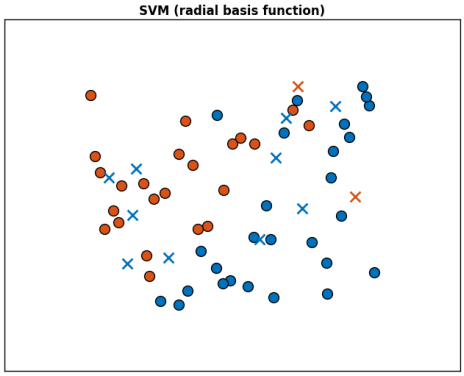
Scenario 3	
$\varepsilon_{\text{train}}$	19.4%
ε_{CV}	28.9%
$\varepsilon_{\text{test}}$	14.5%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	32	5
	T2D	4	21



Scenario 3	
$\varepsilon_{\text{train}}$	1.6%
ε_{CV}	25.8%
$\varepsilon_{\text{test}}$	19.4%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	27	10
	T2D	2	23



Scenario 3	
$\varepsilon_{\text{train}}$	9.7%
ε_{CV}	25.5%
$\varepsilon_{\text{test}}$	19.4%

		Predizione	
		IGT	**T2D**
Realtà	**IGT**	27	10
	T2D	2	23

Tavola 13: Risultati con feature selection basata su expert knowledge per SVM lineare (in alto), SVM con kernel polinomiale (al centro) e SVM con radial basis function (in basso).

Appendice B

Codice per il calcolo degli indici

B.1 Function: `statistical.m`

```
function [gv_ind, gv_name] = statistical(cgm)
% STATISTICAL Computes GV indices based on the statistical
    properties of
% the entire CGM curve.
%
% Indices:
%   1. Mean
%   2. SD: standard deviation
%   3. CV: coefficient of variation (%)
%   4. Median
%   5. Range
%   6. IQR: interquartile range (75%-25%)
%   7. J-index
%
% INPUT arguments:
%   cgm = CGM data (consecutive samples)
%
% OUTPUT arguments:
%   gv_ind = column vector of GV indices
%   gv_name = cell array containing the names of the evaluated
    indices in
%             the order they're stored in gv_ind

%% Initialization
gv_name = {'Mean'
           'SD'
           'CV'
           'Median'
           'Range'
           'IQR'
           'J-index'};

%% 1. Mean
mu = mean(cgm);

%% 2. SD
sigma = std(cgm);

```

```
%% 3. CV
CV = 100 * sigma/mu;

%% 4. Median
med = median(cgm);

%% 5. Range
range = max(cgm) - min(cgm);

%% 6. IQR
IQR = iqr(cgm);

%% 7. J-Index
J = .001 * (mu + sigma)^2;

%% Return variable
gv_ind = [mu
          sigma
          CV
          med
          range
          IQR
          J];

end
```

B.2 Function: `interday.m`

```
function [gv_ind, gv_name] = interday(cgm, day_start, Ndays)
% INTERDAY Computes GV indices based on the variation of the statistical
% properties of the CGM timeseries between days.
%
% Indices:
%   1. SDw: mean of within-day SD
%   2. SDdm: SD of within-day means
%
% INPUT arguments:
%  MANDATORY
%   cgm = CGM data (consecutive samples)
%   day_start = each value within the vector is the index at which each day
%               starts
%  OPTIONAL
%   Ndays = number of days spanned by the measurement
%
% OUTPUT arguments:
%   gv_ind = column vector of GV indices
%   gv_name = cell array containing the names of the evaluated indices in
%             the order they're stored in gv_ind

%% Input arguments handling
if nargin < 3
    Ndays = length(day_start);
end

%% Initialization
gv_name = {'SDw'
           'SDdm'};
```

```
30
31 %% Within-day mean and SD.
32 [mean_within, sd_within] = deal(zeros(Ndays, 1));
33
34 for i = 1:Ndays
35     switch i
36         case Ndays
37             cgm_day = cgm(day_start(i):end);
38         otherwise
39             cgm_day = cgm(day_start(i):day_start(i+1)-1);
40     end
41
42     mean_within(i) = mean(cgm_day);
43     sd_within(i) = std(cgm_day);
44
45 end
46
47 %% 1. SDw
48 SDw = mean(sd_within);
49
50 %% 2. SDdm
51 % Note: in the text the biased estimator is used: CHECK.
52 SDdm = std(mean_within);
53
54 %% Return variable
55 gv_ind = [SDw
56           SDdm];
57
58 end
```

B.3 Function: `target.m`

```
1  function [gv_ind, gv_name] = target(cgm, HYPO, HYPER, n)
2  % TARGET Computes GV indices based on the percentage of time spent
       within,
3  % below and above some predetermined target range.
4  %
5  % Indices:
6  %   1. % of time spent within target
7  %   2. % of time spent below target
8  %   3. % of time spent above target
9  %
10 % INPUT arguments:
11 %  MANDATORY
12 %   cgm = CGM data (consecutive samples)
13 %
14 %  OPTIONAL
15 %   HYPO = hypoglycemic threshold
16 %   HYPER = hyperglycemic threshold
17 %   n = number of samples
18 %
19 % OUTPUT arguments:
20 %   gv_ind = column vector of GV indices
21 %   gv_name = cell array containing the names of the evaluated
       indices in
22 %             the order they're stored in gv_ind
23
24 %% Input arguments handling
25 switch nargin
26     case 1
```

```
        HYPO = 70;
        HYPER = 180;
        n = length(cgm);
    case 2
        HYPER = 180;
        n = length(cgm);
    case 3
        n = length(cgm);
end

%% Initialization
gv_name = {'BG within target'
           'BG below target'
           'BG above target'};

%% 2. % below target
below = 100 * sum(cgm < HYPO) / n;

%% 3. % above target
above = 100 * sum(cgm > HYPER) / n;

%% 1. % within target
% As a fraction in the text: check
within = 100 - (above + below);

%% Return variable
gv_ind = [within
          below
          above];

end
```

B.4 Function: `interpolated.m`

```
function [gv_ind, gv_name] = ...
    interpolated(cgm, t, CONGA_ORD, n)
% INTERDAY Computes GV indices requiring an interpolation of the CGM
% timeseries in order to assess differences between measurements
%
% Indices:
%   1. CONGA
%   2. MODD: mean of daily differences
%
% INPUT arguments:
%  MANDATORY
%   cgm = CGM data (consecutive samples)
%   t = CGM sampling grid
%  OPTIONAL
%   n = number of samples
%   CONGA_ORD = order of the CONGA index (number of hours in the past it
%               refers to)
%
% OUTPUT arguments:
%   gv_ind = column vector of GV indices (-1 if not computed)
%   gv_name = cell array containing the names of the evaluated indices in
%             the order they're stored in gv_ind

```

```
24 %% Input arguments handling
25 switch nargin
26     case 2
27         n = length(cgm);
28         CONGA_ORD = 4;
29     case 3
30         n = length(cgm);
31 end
32
33 %% Initialization
34 gv_name = {'CONGA'
35            'MODD'};
36
37 %% Interpolation of the data on a virtual grid with step = 1 minute
       .
38 % Arguably, it could be done better.
39 step = datenum([0 0 0 0 1 0]); % 1 minute step
40 tv = t(1):step:t(end);
41 cgmv = interp1(t, cgm, tv);
42
43 %% Vector building
44 delay = CONGA_ORD * 60 * step;
45 yesterday = 1440 * step;
46
47 Dc = [];
48 Dm = [];
49
50 for i = 2:n % some more iterations could be avoided
51
52     % Find the index referring to CONGA_ORD hours ago
53     j = find(tv <= t(i) - delay, 1, 'last'); % <= just to be safe
54
55     if ~isempty(j) % if there is actually a meaningful sample in
           cgmv(j)
56         Dc = [Dc
57               cgm(i) - cgmv(j)];
58     end
59
60     % Find the index referring to the same time yesterday
61     j = find(tv <= t(i) - yesterday, 1, 'last'); % <= just to be
           safe
62
63     if ~isempty(j) % if there is actually a meaningful sample in
           cgmv(j)
64         Dm = [Dm
65               abs(cgm(i) - cgmv(j))];
66     end
67
68 end
69
70 %% 1. CONGA
71 if ~isempty(Dc)
72     CONGA = std(Dc);
73 else
74     CONGA = -1;
75 end
76
77 %% 2. MODD
78 if ~isempty(Dm)
79     MODD = mean(Dm);
80 else
81     MODD = -1;
```

```
82 end
83
84 %% Return variable
85 gv_ind = [CONGA
86           MODD];
87
88 end
```

B.5 Function: mage.m

```
1  function [gv_ind, gv_name] = mage(cgm, day_start, Ndays, EF_TH)
2  % MAGE Computes the Mean Amplitude of Glycemic Excursions index
       together
3  % with the complementary Excursion Frequency index.
4  %
5  % Indices:
6  %   1. MAGE
7  %   2. MAGE+
8  %   3. MAGE-
9  %   4. EF: Excursion Frequency (number of excursions > EF_TH)
10 %
11 % INPUT arguments:
12 %  MANDATORY
13 %   cgm = CGM data (consecutive samples)
14 %   day_start = each value within the vector is the index at which
       each day
15 %               starts
16 %
17 %  OPTIONAL
18 %   Ndays = number of days spanned by the measurement
19 %   EF_TH = threshold for an excursion to be significant
20 %
21 % OUTPUT arguments:
22 %   gv_ind = column vector of GV indices
23 %   gv_name = cell array containing the names of the evaluated
       indices in
24 %             the order they're stored in gv_ind
25
26 %% Input arguments handling
27 switch nargin
28     case 2
29         Ndays = length(day_start);
30         EF_TH = 75;
31     case 3
32         EF_TH = 75;
33 end
34
35 %% Initialization
36 gv_name = {'MAGE'
37            'MAGE+'
38            'MAGE-'
39            'EF'};
40
41 %% 1. MAGE
42 [MAGEday_plus, MAGEday_minus, EFday] = deal(zeros(Ndays,1));
43 for d = 1:Ndays
44
45     %% Step 0. Parameters
46     switch d
47         case Ndays
```

```
48              cgmday = cgm(day_start(d):end);
49          otherwise
50              cgmday = cgm(day_start(d):day_start(d+1)-1);
51      end
52
53      nday = length(cgmday);
54      SD_within = std(cgmday);
55
56      %% Step 1. Turning points are only the local extrema
57      [~, i_max] = findpeaks(cgmday);
58      [~, i_min] = findpeaks(-cgmday);
59
60      i_turning = union([1; nday; i_max], i_min);
61      turning = cgmday(i_turning);
62      Nturning = length(i_turning);
63
64      %% Step 2. Turning points of no interest are removed
65      % A turning point is removed if it's not significantly
            different from
66      % BOTH its left and right-hand side RETAINED neighbours.
67
68      to_be_deleted = false(Nturning, 1);
69      for i = 2:Nturning-1    % First and last samples are retained
70
71          condition1 = abs(turning(i)-turning(i-1)) < SD_within;
72          condition2 = abs(turning(i+1)-turning(i)) < SD_within;
73          deletion_condition = condition1 && condition2;
74
75          to_be_deleted(i) = deletion_condition;
76
77      end
78      i_turning = i_turning(~to_be_deleted);
79
80      %% Step 3. Turning points are removed again or moved
            appropriately.
81      % Turning points in the middle of a 3-points monotonic sequence
            are
82      % removed, while turning points that would be retained this way
            are
83      % moved to the closest appropriate local extrema.
84
85      i = 2; % Counter
86      while i < length(i_turning)
87
88          prev = i_turning(i-1);
89          curr = i_turning(i);
90          next = i_turning(i+1);
91
92          prev_slope = cgmday(curr) - cgmday(prev);
93          next_slope = cgmday(next) - cgmday(curr);
94
95          if prev_slope < 0 && next_slope > 0 % Minimum
96
97              % The actual current turning point is the min in the
                    interval
98              [~, temp] = min(cgmday(prev:next));
99              curr = prev+temp-1;
100             i_turning(i) = curr;
101
102             % The actual previous turning point is the max to the
                    left of
103             % the current turning point.
```

```
104             [~, temp] = max(cgmday(prev:curr-1));
105             i_turning(i-1) = prev+temp-1;
106 
107             % The actual following turning point is the max to the
                    right of
108             % the current turning point.
109             [~, temp] = max(cgmday(curr+1:next));
110             i_turning(i+1) = curr+temp;
111 
112             i = i + 1;
113 
114         elseif prev_slope > 0 && next_slope < 0 % Maximum
115 
116             % The actual current turning point is the max in the
                    interval
117             [~, temp] = max(cgmday(prev:next));
118             curr = prev+temp-1;
119             i_turning(i) = curr;
120 
121             % The actual previous turning point is the min to the
                    left of
122             % the current turning point.
123             [~, temp] = min(cgmday(prev:curr-1));
124             i_turning(i-1) = prev+temp-1;
125 
126             % The actual following turning point is the min to the
                    right of
127             % the current turning point.
128             [~, temp] = min(cgmday(curr+1:next));
129             i_turning(i+1) = curr+temp;
130 
131 
132             i = i + 1;
133 
134         else % Middle point
135             % Just remove the turning point
136             i_turning(i) = [];
137         end
138 
139     end
140 
141     %% Step 4. Remove residual spurious turning points
142     % Turning points not significantly different from EITHER
            neighbour are
143     % removed. Some extra processing is needed for the first and
            last sample.
144 
145     % First sample processing
146     sample1 = cgmday(i_turning(1));
147     sample2 = cgmday(i_turning(2));
148 
149     if abs(sample2 - sample1) < SD_within
150         i_turning(1) = [];
151     end
152 
153     % Last sample processing;
154     sample1 = cgmday(i_turning(end-1));
155     sample2 = cgmday(i_turning(end));
156 
157     if abs(sample2 - sample1) < SD_within
158         i_turning(end) = [];
159     end
```

```
160
161     turning = cgmday(i_turning);
162     Nturning = length(i_turning);
163
164     % Internal points
165     to_be_deleted = false(Nturning, 1);
166     for i = 2:Nturning-1
167
168         condition1 = abs(turning(i)-turning(i-1)) < SD_within;
169         condition2 = abs(turning(i+1)-turning(i)) < SD_within;
170         deletion_condition = condition1 || condition2;
171
172         to_be_deleted(i) = deletion_condition;
173
174     end
175
176     i_turning = i_turning(~to_be_deleted);
177     turning = cgmday(i_turning);
178
179 %     %% Test plots
180 %     figure
181 %     hold on
182 %     title('Final turning points')
183 %     plot(cgmday, '.-')
184 %     plot(i_turning, turning, '-o')
185
186     %% Step 5. Compute daily MAGE (plus and minus)
187     excursions = diff(turning);
188     MAGEday_plus(d) = mean(excursions(excursions>0));
189     MAGEday_minus(d) = mean(excursions(excursions<0));
190
191     %% Daily excursion frequency
192     EFday(d) = sum(abs(excursions) > EF_TH);
193
194 end
195
196 %% 2/3. MAGE+ and MAGE-
197 MAGEday_plus(isnan(MAGEday_plus)) = 0; % Correct for 'mean'
        behaviour
198 MAGEday_minus(isnan(MAGEday_minus)) = 0;
199 MAGE_plus = mean(MAGEday_plus);
200 MAGE_minus = -mean(MAGEday_minus);
201
202 MAGE = (MAGE_plus+MAGE_minus)/2;
203
204 % Who knows what the unocmmented code is doing
205
206 %% 4. Excursion Frequency
207 EF = sum(EFday) / Ndays;
208
209 %% Return variable
210 gv_ind = [MAGE
211           MAGE_plus
212           MAGE_minus
213           EF];
214
215 end
```

B.6 Function: `mr.m`

```
function [gv_ind, gv_name] = mr(cgm, R)
% MR Computes the M-value index, based on the average of a non
    linear
% transformation of the CGM data.
%
% Index
%   1. MR, M-value
%
% INPUT arguments:
%   cgm = CGM data (consecutive samples)
%   R = reference value for the transformation
%
% OUTPUT arguments:
%   gv_ind = column vector of GV indices
%   gv_name = cell array containing the names of the evaluated
    indices in
%             the order they're stored in gv_ind

%% Initialization
gv_name = {['MR (R = ' num2str(R) ')']};

%% 1. MR
transformed_cgm = 1000 * abs(log10(cgm/R)).^3;
MR = mean(transformed_cgm);

%% Return variable
gv_ind = MR;

end
```

B.7 Function: grade.m

```
function [gv_ind, gv_name] = grade(cgm, HYPO, HYPER)
% GRADE Computes Hill's GRADE index.
%
% Indices:
%   1. GRADE
%   1. % GRADE due to euglycemia
%   2. % GRADE due to hypoglycemia
%   3. % GRADE due to hyperglycemia
%
% INPUT arguments:
%  MANDATORY
%   cgm = CGM data (consecutive samples)
%
%  OPTIONAL
%   HYPO = hypoglycemic threshold
%   HYPER = hyperglycemic threshold
%
% OUTPUT arguments:
%   gv_ind = column vector of GV indices
%   gv_name = cell array containing the names of the evaluated
    indices in
%             the order they're stored in gv_ind

%% Input arguments handling
switch nargin
    case 1
        HYPO = 70;
        HYPER = 180;
```

```
28      case 2
29          HYPER = 180;
30  end
31
32  %% Initialization
33  gv_name = {'GRADE'
34             'GRADEeu'
35             'GRADEhypo'
36             'GRADEhyper'};
37
38  %% GRADE
39  % Wrong in the paper? x*18 VS x/18
40  GRADE = 425 * (log10(log10(cgm/18)) + .16).^2;
41  GRADEmean = mean(GRADE);
42  G_tot = sum(GRADE);
43
44  %% 3. % below target
45  GRADEhypo = 100 * sum(GRADE(cgm < HYPO)) / G_tot;
46
47  %% 4. % above target
48  GRADEhyper = 100 * sum(GRADE(cgm > HYPER)) / G_tot;
49
50  %% 2. % within target
51  GRADEeu = 100 - (GRADEhypo + GRADEhyper);
52
53  %% Return variable
54  gv_ind = [GRADEmean
55            GRADEeu
56            GRADEhypo
57            GRADEhyper];
58
59  end
```

B.8 Function: rodbard.m

```
1   function [gv_ind, gv_name] = rodbard(cgm, HYPO, HYPER, ABCD, n)
2   % RODBARD Computes GV indices based on simple non linear
        transformations of
3   % the CGM timeseries.
4   %
5   % Indices:
6   %   1. Hypo Index
7   %   2. Hyper Index
8   %   3. IGC: Index of Glycemic control
9   %
10  % INPUT arguments:
11  %  MANDATORY
12  %   cgm = CGM data (consecutive samples)
13  %
14  %  OPTIONAL
15  %   HYPO = hypoglycemic threshold
16  %   HYPER = hyperglycemic threshold
17  %   ABCD = vector containing the parameters [a b c d]'
18  %   n = number of samples
19  %
20  % OUTPUT arguments:
21  %   gv_ind = column vector of GV indices
22  %   gv_name = cell array containing the names of the evaluated
        indices in
23  %             the order they're stored in gv_ind
```

```
24
25 %% Input arguments handling
26 switch nargin
27     case 1
28         HYPO = 70;
29         HYPER = 180;
30         ABCD = [1.1 2 30 30]';
31         n = length(cgm);
32     case 2
33         HYPER = 180;
34         ABCD = [1.1 2 30 30]';
35         n = length(cgm);
36     case 3
37         ABCD = [1.1 2 30 30]';
38         n = length(cgm);
39     case 4
40         n = length(cgm);
41 end
42
43 %% Initialization
44 gv_name = {'Hypo Index'
45            'Hyper Index'
46            'IGC'};
47
48 %% Notation consistency
49 a = ABCD(1);
50 b = ABCD(2);
51 c = ABCD(3);
52 d = ABCD(4);
53 LLTR = HYPO;
54 ULTR = HYPER;
55
56 %% 1. Hypo Index
57 % Wrong definition in the text (summation).
58 hypoI = sum((LLTR - cgm(cgm < LLTR)).^b) / (n*d);
59
60 %% 2. Hyper Index
61 % Wrong definition in the text (summation).
62 hyperI = sum((cgm(cgm > ULTR) - ULTR).^a) / (n*c);
63
64 %% 3. IGC
65 IGC = hypoI + hyperI;
66
67 %% Return variable
68 gv_ind = [hypoI
69           hyperI
70           IGC];
71
72 end
```

B.9 Function: `kovatchev.m`

```
1 function [gv_ind, gv_name] = kovatchev(cgm, day_start, Ndays, K_TH,
      KPAR)
2 % KOVATCHEV Computes GV indices based on  Kovatchev's
    symmetrization of the
3 % blood glucose scale.
4 %
5 % Indices:
6 %   1. LBGI: Low Blood Glucose Index
```

```
7  %   2. HBGI: High Blood Glucose Index
8  %   3. BGRI: Blood Glucose Risk Index
9  %   4. ADRR: Average Daily Risk Range
10 %
11 % INPUT arguments:
12 %  MANDATORY
13 %   cgm = CGM data (consecutive samples)
14 %   day_start = each value within the vector is the index at which
       each day
15 %               starts
16 %
17 %  OPTIONAL
18 %   Ndays = number of days spanned by the measurement
19 %   K_TH = threshold used in the distinction between risk of hypo
       and
20 %          hyperglycemia
21 %   ABCD = vector containing the parameters [alpha beta gamma]'
       needed for
22 %          the symmetrization function
23 %
24 % OUTPUT arguments:
25 %   gv_ind = column vector of GV indices
26 %   gv_name = cell array containing the names of the evaluated
       indices in
27 %             the order they're stored in gv_ind
28
29 %% Input arguments handling
30 switch nargin
31     case 2
32         Ndays = length(day_start);
33         K_TH = 112.5;
34         KPAR = [1.084 5.381 1.509]';
35     case 3
36         K_TH = 112.5;
37         KPAR = [1.084 5.381 1.509]';
38     case 4
39         KPAR = [1.084 5.381 1.509]';
40 end
41
42 %% Initialization
43 gv_name = {'LBGI'
44            'HBGI'
45            'BGRI'
46            'ADRR'};
47
48 %% Notation consistency
49 alpha = KPAR(1);
50 beta = KPAR(2);
51 gamma = KPAR(3);
52
53 %% Symmetrization
54 f = gamma * ((log(cgm)).^alpha - beta);
55
56 %% Risk
57 [rh, rl] = deal(10 * f.^2);
58 rl(cgm > K_TH) = 0;
59 rh(cgm < K_TH) = 0;
60
61 %% 1. LBGI
62 LBGI = mean(rl);
63
64 %% 2. HBGI
```

```
HBGI = mean(rh);

%% 3. BGRI
BGRI = LBGI+HBGI;

%% 4. ADRR
ADRRday = zeros(Ndays, 1);

for j = 1:Ndays

    switch j
        case Ndays
            rlday = rl(day_start(j):end);
            rhday = rh(day_start(j):end);
        otherwise
            rlday = rl(day_start(j):day_start(j+1)-1);
            rhday = rh(day_start(j):day_start(j+1)-1);
    end

    ADRRday(j) = max(rlday) + max(rhday);

end

ADRR = mean(ADRRday);

%% Return variable
gv_ind = [LBGI
          HBGI
          BGRI
          ADRR];

end
```

B.10 Function: `moments.m`

```
function [gv_ind, gv_name] = moments(cgm, t)
% INTERDAY Computes GV indices based on invariant moments of the binary
% image used in computing the area under the CGM curve.
%
% Indices:
%   1-7. phi(1-7): Invariant image moments (Hu 1962)
%
% INPUT arguments:
%   cgm = CGM data (consecutive samples)
%   t = CGM sampling grid
%
% OUTPUT arguments:
%   gv_ind = column vector of GV indices (-1 if not computed)
%   gv_name = cell array containing the names of the evaluated indices in
%             the order they're stored in gv_ind

%% Initialization
gv_name = {'phi1'
           'phi2'
           'phi3'
           'phi4'
           'phi5'
           'phi6'
```

```
24              'phi7'};
25
26  %% Conversion to image
27  % The binary image used is = 1 between the CGM curve and a straight
        line
28  % fixed at the minimum BG level measured.
29
30  % Image size
31  timespan = daysact(t(1), t(end)) * 1440; % minutes
32  N = round(1.1*timespan); % #columns
33
34  top = max(cgm);
35  bottom = min(cgm);
36  cgmspan = top - bottom;
37  M = round(1.1*cgmspan); % #rows
38
39  % Time in terms of pixels
40  x = (t - t(1)) / (t(end) - t(1)) * N;
41  x = round(x);
42
43  % Cgm in terms of pixels (images start from the top)
44  y = top - cgm;
45  y = round(y);
46
47  % Appending polygon edges
48  x = [x(1); x; x(end)];
49  y = [cgmspan; y; cgmspan];
50
51  % Creating the actual image
52  I = poly2mask(x, y, M, N);
53  I = double(I);
54
55  % % Image check
56  % figure
57  % imagesc(I)
58  % colormap gray
59  % colorbar
60
61  %% Mask: x and y coordinates of each point where I == 1
62  [xv, yv] = find(I);
63
64  %% Area
65  A = length(xv);
66
67  %% Centroid
68  xc = mean(x);
69  yc = mean(y);
70
71  %% Central normalized moments of interest
72  p = [2 1 0 3 2 1 0]';
73  q = [0 1 2 0 1 2 3]';
74  gamma = (p+q)/ 2 + 1;
75
76  Npq = length(p);
77  eta = nan(Npq, 1);
78
79  for i = 1:Npq
80
81      temp = (xv-xc).^p(i) .* (yv-yc).^q(i);
82      eta(i) = sum(temp) / A^gamma(i);
83
84  end
```

```
85
86  %% Rename the moments for the sake of convenience
87  for i = 1:Npq
88
89      eval(['eta' num2str(p(i)) num2str(q(i)) ' = ' ...
90            'eta(' num2str(i) ');'])
91
92  end
93
94
95  %% 1-7. Invariant image moments
96  phi1 = eta20 - eta02;
97
98  phi2 = (eta20 - eta02)^2 + 4*eta11^2;
99
100 phi3 = (eta30 - 3*eta12)^2 + (eta03 - 3*eta21)^2;
101
102 phi4 = (eta30 + eta12)^2 + (eta03 +eta21)^2;
103
104 phi5 = (eta30 - 3*eta12) * (eta30 + eta12) * ...
105             ((eta30 + eta12)^2 - 3*(eta21 + eta03)^2) + ...
106        (3*eta21 - eta03) * (eta21 + eta03) * ...
107             (3*(eta30 + eta12)^2 - (eta21 + eta03)^2);
108
109 phi6 = (eta20 - eta02) * ((eta30 + eta12)^2 - (eta21 + eta03)^2) +
        ...
110             4*eta11 * (eta30 + eta12) * (eta21 + eta03);
111
112 phi7 = (3*eta21 - eta03) * (eta30 + eta12) * ...
113             ((eta30 + eta12)^2 - 3*(eta21 + eta03)^2) + ...
114        (3*eta12 - eta30) * (eta21 + eta03) * ...
115             (3*(eta30 + eta12)^2 - (eta21 + eta03)^2);
116
117
118 %% Return variable
119 gv_ind = [phi1
120           phi2
121           phi3
122           phi4
123           phi5
124           phi6
125           phi7];
126
127 end
```

Appendice C

Codice per la classificazione

C.1 Implementazione generale: `GVClassifier.py`

```
# -*- coding: utf-8 -*-
"""
Sample GV-based classifier: linear SVM
"""
#============================================================
# Import
#============================================================
import numpy as np
import gvutil as gv
import matplotlib.pyplot as plt
from sklearn.cross_validation import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.grid_search import GridSearchCV
from sklearn import preprocessing
from sklearn import svm # Import the appropriate classifier

#============================================================
# Parameters
#============================================================
SCENARIO = 0 # Choose a scenario: (0, 1, 2, 3)
K = 4        # Set the number of folds for CV

#============================================================
# Data loading: build appropriate training and test set
#============================================================
(Xtrain, Xtest, Ltrain, Ltest, Xvars, patients, m, n) = \
                                    gv.LoadScenario(SCENARIO)

#============================================================
# Appropriate label format
#============================================================
Ytrain = Ltrain-1 # Labels = (0, 1)

#============================================================
# Scaler (standardization)
#============================================================
z_norm_scaler = preprocessing.StandardScaler()

#============================================================
# Classifier (e.g. Support Vector Machine)
```

```
#================================================================
sv = svm.SVC(C=1.0, # Default (it will be set later)
             kernel='linear',
             probability=False,
             tol=1e-4,
             max_iter=-1)

#================================================================
# Build the pipeline
#================================================================
sequence = [('norm', z_norm_scaler),
            ('svm', sv)]

clf = Pipeline(sequence)

#================================================================
# Grid search for optimal parameters
#================================================================
parameters = dict(svm__C = np.arange(.0001, 10, .001))

clf = GridSearchCV(clf,
                   parameters,
                   cv=K,
                   verbose=0)
#================================================================
# Training
#================================================================
clf.fit(Xtrain, Ytrain)

Ytrain_hat = clf.predict(Xtrain)
Ltrain_hat = Ytrain_hat + 1

train_error = 1 - accuracy_score(Ltrain, Ltrain_hat)

#================================================================
# Testing
#================================================================
Ytest_hat = clf.predict(Xtest)
Ltest_hat = Ytest_hat + 1

test_error = 1 - accuracy_score(Ltest, Ltest_hat)

#================================================================
# Cross-validation
#================================================================
fold_score = cross_val_score(clf, Xtrain, Ytrain,
                             cv=K,
                             scoring='accuracy')

fold_error = 1 - fold_score
cv_error = fold_error.mean()
cv_sd = fold_error.std()

#================================================================
# Output
#================================================================
# ...
```

Bibliografia

[1] D. L. Nelson and M. M. Cox, *Lehninger Principles of Biochemistry, Fourth Edition*. Freeman, 4th ed., 2004.

[2] V. Monesi, *Istologia*. Piccin, 6th ed., 2012.

[3] "Diagnosis and classification of diabetes mellitus," *Diabetes Care*, vol. 36, no. SUPPL.1, pp. 67–74, 2013.

[4] "Standards of medical care in diabetes - 2013," *Diabetes Care*, vol. 36, no. Supplement 1, pp. S11–S66, 2012.

[5] E. M. Benjamin, "Self-Monitoring of Blood Glucose: The Basics," *Clinical Diabetes*, vol. 20, no. 1, pp. 45–47, 2002.

[6] D. C. Klonoff, B. Buckingham, J. S. Christiansen, V. M. Montori, W. V. Tamborlane, R. A. Vigersky, and H. Wolpert, "Continuous glucose monitoring: An endocrine society clinical practice guideline," *Journal of Clinical Endocrinology and Metabolism*, vol. 96, no. 10, pp. 2968–2979, 2011.

[7] J. Kropff and J. H. DeVries, "Continuous Glucose Monitoring, Future Products, and Update on Worldwide Artificial Pancreas Projects.," *Diabetes Technology & Therapeutics*, vol. 18 Suppl 2, pp. S253–63, 2016.

[8] J. E. Lane, J. P. Shivers, and H. Zisser, "Continuous glucose monitors: current status and future developments.," *Current Opinion in Endocrinology, Diabetes, and Obesity*, vol. 20, no. 2, pp. 106–111, 2013.

[9] S. P. Nichols, A. Koh, W. L. Storm, J. H. Shin, and M. H. Schoenfisch, "Biocompatible materials for continuous glucose monitoring devices," *Chemical Reviews*, vol. 113, pp. 2528–2549, 2013.

[10] S. Guerra, A. Facchinetti, G. Sparacino, G. De Nicolao, and C. Cobelli, "Enhancing the accuracy of subcutaneous glucose sensors: A real-time deconvolution-based approach," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 6, pp. 1658–1669, 2012.

[11] G. Freckmann, S. Pleus, M. Link, E. Zschornack, H. Klötzer, and C. Haug, "Performance evaluation of three continuous glucose monitoring systems: comparison of six sensors per subject in parallel.," *Journal of diabetes science and technology*, vol. 7, no. 4, pp. 842–53, 2013.

[12] T. S. Bailey, A. Chang, and M. Christiansen, "Clinical Accuracy of a Continuous Glucose Monitoring System With an Advanced Algorithm," *Journal of Diabetes Science and Technology*, vol. 9, no. 2, pp. 209–214, 2015.

[13] M. Christiansen, T. Bailey, E. Watkins, D. Liljenquist, D. Price, K. Nakamura, R. Boock, and T. Peyser, "A new-generation continuous glucose monitoring system: improved accuracy and reliability compared with a previous-generation system.," *Diabetes Technology & Therapeutics*, vol. 15, no. 10, pp. 881–8, 2013.

[14] R. M. Bergenstal, "Glycemic variability and diabetes complications: Does it matter? simply put, there are better glycemic markers!," *Diabetes Care*, vol. 38, no. 8, pp. 1615–1621, 2015.

[15] I. B. Hirsch, "Glycemic variability and diabetes complications: Does it matter? of course it does!," *Diabetes Care*, vol. 38, no. 8, pp. 1610–1614, 2015.

[16] B. Kovatchev and C. Cobelli, "Glucose Variability: Timing, Risk Analysis, and Relationship to Hypoglycemia in Diabetes," *Diabetes Care*, vol. 39, no. April, pp. 502–510, 2016.

[17] D. Rodbard, "The challenges of measuring glycemic variability.," *Journal of Diabetes Science and Technology*, vol. 6, no. 3, pp. 712–5, 2012.

[18] C. Fabris, A. Facchinetti, G. Sparacino, M. Zanon, S. Guerra, A. Maran, and C. Cobelli, "Glucose Variability Indices in Type 1 Diabetes: Parsimonious Set of Indices Revealed by Sparse Principal Component Analysis," *Diabetes Technology & Therapeutics*, vol. 16, no. 10, pp. 11–13, 2014.

[19] J. M. Wojcicki, "J-index. a new proposition of the assessment of current glucose control in diabetic patients," *Hormone and Metabolic Research*, vol. 27, no. 01, pp. 41–42, 1995.

[20] D. Rodbard, "New and Improved Methods to Characterize Glycemic Variability Using Continuous Glucose Monitoring," *Diabetes Technology & Therapeutics*, vol. 11, no. 9, 2009.

[21] C. M. McDonnell, S. M. Donath, S. I. Vidmar, G. A. Werther, and F. J. Cameron, "A novel approach to continuous glucose analysis utilizing glycemic variation," *Diabetes Technology & Therapeutics*, vol. 7, no. 2, pp. 253–263, 2005.

[22] F. J. Service, G. D. Molnar, J. W. Rosevear, E. Ackerman, L. C. Gatewood, and W. F. Taylor, "Mean amplitude of glycemic excursions, a measure of diabetic instability," *Diabetes*, vol. 19, no. 9, pp. 644–655, 1970.

[23] P. A. Baghurst, "Calculating the mean amplitude of glycemic excursion from continuous glucose monitoring data: an automated algorithm," *Diabetes Technology & Therapeutics*, vol. 13, no. 3, pp. 296–302, 2011.

[24] M. Wiley, R. Bunescu, C. Marling, J. Shubrook, and F. Schwartz, "Automatic detection of excessive glycemic variability for diabetes management," *Proceedings - 10th International Conference on Machine Learning and Applications, ICMLA 2011*, vol. 2, pp. 148–154, 2011.

[25] C. R. Marling, J. H. Shubrook, S. J. Vernier, M. T. Wiley, and F. L. Schwartz, "Characterizing blood glucose variability using new metrics with continuous glucose monitoring data," *Journal of Diabetes Science and Technology*, vol. 5, no. 4, pp. 871–8, 2011.

[26] C. Marling, M. Wiley, R. Bunescu, J. Shubrook, and F. Schwartz, "Intelligent Diabetes Management," *AI Magazine*, vol. 33, no. 2, pp. 67–78, 2012.

[27] N. R. Hill, P. C. Hindmarsh, R. J. Stevens, I. M. Stratton, J. C. Levy, and D. R. Matthews, "A method for assessing quality of control from glucose profiles," *Diabetic Medicine*, vol. 24, no. 7, pp. 753–758, 2007.

[28] B. P. Kovatchev, D. J. Cox, L. A. Gonder-Frederick, and W. Clarke, "Symmetrization of the Blood Glucose Measurement Scale and Its Applications," *Diabetes Care*, vol. 20, no. 11, pp. 1655–1658, 1997.

[29] M. Hu, "Visual pattern recognition by moment invariants," *IRE transactions on information theory*, vol. 8, no. 2, pp. 179–187, 1962.

[30] N. W. Struble, "Measuring Glycemic Variability and Predicting Blood Glucose Levels Using Machine Learning Regression Models," Master's thesis, Russ College of Engineering and Technology of Ohio University, 12 2013.

[31] F. Hlawatsch and G. F. Boudreaux-Bartels, "Linear and quadratic time-frequency signal representations," *IEEE Signal Processing Magazine*, vol. 9, no. 2, pp. 21–67, 1992.

[32] C. Fabris, *Glucose Variability Assessment in Diabetes Mellitus Monitoring and Control*. PhD thesis, Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Padova, 2015. http://paduaresearch.cab.unipd.it/7986/.

[33] V. Bettinardi, "Classificazione della qualità del controllo glicemico in soggetti affetti da diabete di tipo 1," Master's thesis, Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Padova, 2015. http://tesi.cab.unipd.it/47990/.

[34] P. Augstein, P. Heinke, L. Vogt, R. Vogt, C. Rackow, K. Kohnert, and E. Salzsieder, "Q-Score: development of a new metric for continuous glucose monitoring that enables stratification of antihyperglycaemic therapies," *BMC Endocrine Disorders*, vol. 15, no. 1, p. 22, 2015.

[35] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics, Springer New York, 2009.

[36] X. Wang and J. Tian, "A gene selection method for cancer classification," *Computational and Mathematical Methods in Medicine*, vol. 2012, pp. 389–422, 2012.

[37] F. Sambo, B. Di Camillo, A. Franzin, A. Facchinetti, L. Hakaste, J. Kravic, G. Fico, J. Tuomilehto, L. Groop, R. Gabriel, *et al.*, "A bayesian network

analysis of the probabilistic relations between risk factors in the predisposition to type 2 diabetes," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2119–2122, IEEE, 2015.

[38] L. Groop, C. Forsblom, M. Lehtovirta, T. Tuomi, S. Karanko, M. Nissen, B. O. Ehrnstrom, B. Forsen, B. Isomaa, B. Snickars, and M. R. Taskinen, "Metabolic consequences of a family history of NIDDM (the Botnia study): evidence for sex-specific parental effects," *Diabetes*, vol. 45, no. 11, pp. 1585–1593, 1996.

[39] A. J. Pyykkonen, K. Raikkonen, T. Tuomi, J. G. Eriksson, L. Groop, and B. Isomaa, "Stressful life events and the metabolic syndrome: the prevalence, prediction and prevention of diabetes (PPP)-Botnia Study," *Diabetes Care*, vol. 33, no. 2, pp. 378–384, 2010.

[40] P. Almgren, M. Lehtovirta, B. Isomaa, L. Sarelin, M. R. Taskinen, V. Lyssenko, T. Tuomi, and L. Groop, "Heritability and familiality of type 2 diabetes and related quantitative traits in the Botnia Study," *Diabetologia*, vol. 54, no. 11, pp. 2811–2819, 2011.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[42] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[43] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class AdaBoost," *Statistics and Its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[44] J. H. DeVries, "Glucose variability: Where it is important and how to measure it," *Diabetes*, vol. 62, no. 5, pp. 1405–1408, 2013.

[45] S. E. Siegelaar, F. Holleman, J. B. L. Hoekstra, and J. H. DeVries, "Glucose variability; does it matter?," *Endocrine Reviews*, vol. 31, no. 2, pp. 171–182, 2010.