Università degli studi di padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

Tecniche di Apprendimento Automatico per l'identificazione dell'Iperaldosteronismo Primario

Relatori:

Ch.mo Prof. Andrea Pietracaprina

Ch.mo Prof. Geppino Pucci

Dott. Carlo Fantozzi

Ch.mo Prof. Loris Nanni

Correlatore:

Ch.mo Prof.Dott. GianPaolo Rossi

Laureando:

Nicola Lazzarini

Anno Accademico 2011/2012

"Nothing is particularly hard if you divide it into small jobs"

Henry Ford

Sommario

In questo lavoro di tesi si è cercato di sviluppare un classificatore che permetta l'identificazione dei pazienti affetti da Iperaldosteronismo Primario (PA) dovuto alla presenza dell'adenoma di Conn (APA). Si tratta di una forma di PA guaribile chirurgicamente, la sua corretta identificazione precoce permette di guarire il PA e di correggere l'ipertensione arteriosa. L'esame "principe" per l'individuazione di APA è l'AVS (Adrenal Vein Sampling): si tratta di un esame invasivo, costoso e che richiede buone competenze tecniche per la sua esecuzione. Per quanto detto è importante sottoporvi solamente i pazienti che hanno un'elevata probabilità di essere affetti dalla patologia. L'obiettivo di questo lavoro è stato quello di fornire ai medici un modello matematico che possa aiutarli nel selezionare i pazienti da sottoporre ad AVS. Nel 1998 è stato realizzato un classificatore con le stesse finalità, in quel caso è stata utilizzata la tecnica della Logistic Regression. In questa tesi si sono provati nuovi approcci rispetto a quanto fatto precedentemente, in particolare si sono utilizzati: Probability Estimation Tree (PET) e Support Vector Machine (SVM). Il dataset usato per la fase di training proviene dallo studio PAPY effettuato nel 2006 con lo scopo di definire la prevalenza di PA in Italia. Per la fase di validazione si è ricorsi ad una generalizzazione della Cross Validation: Leave One Out Clinica. Si tratta di un metodo che permette di preservare, in fase di validazione, le diverse caratteristiche dei pazienti dovute alla provenienza da cliniche differenti. L'analisi del modello sviluppato nel 1998 ha dimostrato la sua validità, perciò si è provato a combinare tale classificatore con quanto di nuovo realizzato. In termini di prestazioni le combinazioni citate hanno portato ai risultati migliori. Rispetto al lavoro svolto nel 1998 si sono ottenuti significativi miglioramenti: si è creato un modello, usando i PET, che offre informazioni ai medici riguardo alle variabili che più discriminano tra pazienti con e senza APA. Inoltre in termini di prestazioni, a parità di capacità di individuare i pazienti positivi, i nuovi modelli hanno una probabilità maggiore di riconoscere anche i pazienti negativi rispetto al lavoro precedente.

Indice

Capitolo 1

Introduzione

L'iperaldosteronismo primario (Primary Aldosteronism, PA) è una causa comune, sebbene spesso non diagnosticata, dell'ipertensione arteriosa guaribile. Si tratta di una condizione patologica in cui vi è un'elevata ed inappropriata secrezione di aldosterone che non avviene sotto il normale controllo del sistema renina-angiotensina [1]. Le due forme, o sottotipi, principali di iperaldosteronismo primario sono: *Iperaldosteronismo Bilaterale Idiopatico* (IHA) ed *Adenoma Secernente Aldosterone* (APA). Questa patologia fu descritta per la prima da J.W. Conn nel 1955; infatti in una situazione di APA si parla comunemente di *sindrome di Conn*. Il termine "primario" sottolinea l'ignoranza riguardo ai meccanismi molecolari sottostanti. L'APA è solitamente trattato in maniera chirurgica, mentre l'IHA è trattato con cure farmacologiche.

Il test di screening più popolare per l'identificazione dell'iperaldosteronismo primario è il rapporto aldosterone/renina (ARR). Si tratta di un metodo semplice, anche se rudimentale, che analizza il rapporto tra due valori biochimici. Va tenuto presente che l'ARR è una cruda analisi bivariata il cui valore dipende dalla concentrazione di aldosterone nel plasma (PAC) e dai livelli di renina. Perciò diversi valori di PAC e renina possono portare ad un medesimo rapporto. Ciò significa che tutti i pazienti che hanno una produzione di renina soppressa avranno un aumento di ARR nonostante valori normali di PAC e quindi assenza d'iperaldosteronismo primario. Per questo ed altri motivi vengono utilizzati dei test di conferma per determinare la presenza o meno di PA.

Una volta assicurata l'esistenza della patologia è necessario identificarne il sottotipo poiché, come si è detto, forme diverse hanno trattamenti diversi. Il test ARR non permette di discriminare i pazienti affetti da IHA da quelli con APA, e altre tecniche basate sull'analisi di immagini si sono rivelate allo stesso modo inefficienti. Molti esperti sono concordi nel ritenere che il test "principe" per la discriminazione tra le due forme sia il

[1] È un meccanismo ormonale che regola la pressione sanguigna, il volume plasmatico circolante(volemia) ed il tono della muscolatura arteriosa

cateterismo venoso surrenalico (AVS: Adrenal Vein Sampling). L'AVS è un esame costoso, tecnicamente impegnativo e gravato da un minimo rischio per il paziente (rottura della vena surrenalica) pertanto andrebbe riservato ai pazienti che sembrano mostrare un'elevata probabilità di APA. Gli aspetti negativi dell'AVS hanno portato alla necessità di poter conoscere, con buona precisione, quali pazienti possono essere affetti da APA in modo tale da poter selezionare al meglio i candidati da sottoporre al cateterismo. Per soddisfare questa necessità si è cercato di sviluppare un modello/classificatore che sia in grado di identificare quali pazienti ipertesi sono colpiti da Adenoma di Conn.

Nel 1998 è stato affrontato un problema analogo in [1] dove è stato sviluppato un modello per l'identificazione della sindrome di Conn utilizzando la Regressione Logistica. Sulla base dei valori assunti da quattro variabili (valori biochimici del paziente) è possibile calcolare la probabilità della presenza della patologia. Il progetto ha utilizzato i dati riguardanti i pazienti provenienti dalla Clinica Medica 4 di Padova e dal Dipartimento di Medicina Interna dell'Ospedale di Reggio Emilia. Da questo lavoro nasce un modello facilmente implementabile che offre buone prestazioni in termini di accuratezza, sensibilità e specificità. Le tre metriche di prestazione citate sono molto spesso utilizzate nella valutazione di un classificatore. L'accuratezza indica la percentuale di pazienti che vengono classificati correttamente dal modello, la sensibilità rappresenta la probabilità che una persona malata venga classificata come positiva, mentre la specificità si riferisce alla capacità di riconoscere le persone sane come negative. Il modello permette inoltre di superare i limiti dell'ARR dovuti all'analisi bivariata di un rapporto tra valori. All'interno di questa tesi il modello creato con la Logistic Regression è stato nuovamente validato su un insieme di dati più recenti e diversi rispetto a quelli del 1998. Solitamente i modelli vengono validati usando una tecnica molto comune: la *k-Fold Cross Validation*. In tale validazione i record sono suddivisi in gruppi, in maniera casuale, e ciascuno di essi viene utilizzato *k-1* volte per la creazione del modello ed una volta per la validazione. Le prestazioni finali sono date dalla media delle k singole valutazioni che si ottengono. I dati utilizzati in questo lavoro provengono da cliniche, o centri specializzati, diverse per cui si è ritenuto corretto non considerare nello stesso gruppo pazienti provenienti da sorgenti diverse. Operando in tale modo si tiene conto delle diverse caratteristiche che possono avere record riferiti a centri differenti. A questo proposito si è utilizzata una generalizzazione della Cross Validation nella quale ciascun gruppo contiene pazienti provenienti dal medesimo centro: questo nuovo metodo di validazione ha preso il nome di *Leave One Out Clinica*.

Nel 2006 è stato pubblicato il primo studio prospettico sull'iperaldosteronismo primario: lo studio PAPY (Primary Aldosteronism Prevalence in hYpertensives). Si tratta di un lavoro progettato ad hoc per fornire dati "solidi" sulla reale prevalenza della patologia. A partire dallo studio PAPY è stato realizzato un database che contiene svariate informazioni riguardo i pazienti ipertesi, molte di più rispetto a quelle disponibili nel 1998.

L'obiettivo di questa tesi è stato quello di utilizzare queste nuove informazioni per provare a raggiungere due risultati: (1) cercare di trovare nuove variabili che aiutino ad identificare la presenza di APA nei pazienti, (2) cercare di sviluppare un classificatore che offra delle performance migliori rispetto a quelle raggiunte con la Logistic Regression. L'attenzione si concentra sui malati di APA in quanto la corretta identificazione del sottotipo può portare alla guarigione tramite intervento chirurgico.

Sono stati sviluppati due nuovi modelli utilizzando tecniche di learning diverse: Probability Estimation Tree (PET) e Support Vector Machine (SVM). Gli alberi offrono un modello altamente informativo, di facile comprensione ed utilizzo, ma le prestazioni che si ottengono non sono elevatissime. Risultati migliori si sono raggiunti con SVM che però, a differenza dei PET, non fornisce particolari informazioni sulle variabili e sul loro valore discriminante rispetto alla malattia. Visto che il lavoro svolto nel '98 sembra offrire buoni risultati si è provato a combinare tale modello con quelli sviluppati in questo nuovo progetto. La suddetta combinazione permette di ottenere, in termini di performance, i migliori risultati in assoluto; in particolare le prestazione più interessanti sono frutto dell'aggregazione del modello SVM con quello logistico del '98. Significativi sono i miglioramenti che si ottengono, scegliendo tale combinazione, considerando elevati valori di sensibilità (alta capacità del modello di riconoscere pazienti malati). In questo caso, a parità di sensibilità, vi è un aumento di circa 20 punti percentuali per quanto riguarda il valore della specificità raggiunto nel 1998. I miglioramenti si notano anche considerando solamente il modello SVM, senza aggregazione con quello logistico. In questa situazione vi è un aumento di 10 punti percentuali di specificità a parità di sensibilità fissata a valori elevati. Ciò significa che a parità di capacità nell'individuare le persone malate i nuovi modelli basati su SVM sono migliori nell'identificazione delle persone sane rispetto a quanto accadeva nel 1998. C'è inoltre da considerare che il modello PET offre molte informazioni riguardo alle variabili che sono significativamente importanti per l'identificazione della malattia, ciò avviene in maniera molto limitata nel lavoro precedente.

Di seguito viene presentata la struttura della tesi:

Capitolo 1 Introduzione: viene presentato il lavoro svolto e le motivazioni che hanno portato allo sviluppo di questa tesi. Si accenna al problema medico ed al lavoro simile realizzato nel 1998. Sono presentate le tecniche usate per lo sviluppo dei modelli e gli approcci seguiti per la loro valutazione. Nell'ultima parte vengono presentati gli obiettivi prefissati e vengono illustrati i miglioramenti ottenuti.

Capitolo 2 Strumenti di Apprendimento Automatico: in questo capitolo sono descritti i fondamenti teorici sui quali si basa il lavoro della tesi. Si parte dalla definizione del problema della classificazione, in seguito vengono presentate le tecniche utilizzate per lo sviluppo dei classificatori: PET e SVM. Viene fornito un accenno

sulla Logistic Regression, approccio utilizzato nel lavoro svolto del 1998. Una sezione è invece dedicata alle metriche di prestazione utilizzate per valutare la bontà di un modello. È molto comune avere a che fare con valori mancanti quando si affrontano problemi che utilizzano dati del mondo reale, per questo motivo è presente una sezione che descrive come questi missing values vengono trattati. Infine è descritta la gestione di un problema molto comune in ambito medico: lo sbilanciamento dei dataset, ovvero la presenza molto limitata dei record positivi rispetto all'abbondanza di quelli negativi.

Capitolo 3 Identificazione dell'Iperaldosteronismo Primario: viene introdotto in maniera specifica il problema medico che riguarda questa tesi. È presente una descrizione della patologia e dei suoi vari sottotipi. Un paragrafo viene dedicato alla strategia di screening usata solitamente per l'identificazione dell'iperaldosteronismo primario. In questo capitolo viene descritto il lavoro, svolto nel 1998, che ha avuto il medesimo obiettivo di realizzare un modello per l'identificazione dell'iperaldosteronismo primario. Tale lavoro viene validato sul medesimo dataset utilizzato per la costruzione dei nuovi classificatori. Il dataset citato, chiamato PAPY, è descritto e analizzato alla fine del capitolo.

Capitolo 4 Risultati Raggiunti: Si presentano e si analizzano i modelli sviluppati con le diverse tecniche. Vengono descritti i percorsi che hanno portato alla versione finale dei modelli, si parla inoltre delle tappe intermedie affrontate durante lo sviluppo del progetto. I nuovi classificatori sviluppati sono confrontati rispetto al modello realizzato nel 1998. Date le buone prestazioni del modello logistico, esso viene combinato con quanto realizzato in questo lavoro. Nel capitolo vengono analizzati i modelli ottenuti da tale combinazione. Alla fine viene fatto un confronto generale tra tutti i modelli citati per capire quali siano i risultati raggiunti.

Capitolo 5 Conclusioni: Si conclude la tesi con una sintesi del lavoro svolto ed un commento finale su quanto ottenuto. Vengono riportati i risultati ed i miglioramenti ottenuti rispetto a quanto fatto nel lavoro precedente. Sono fatte delle considerazione sulla base delle possibili necessità presentate dai medici. Sono infine presentati eventuali sviluppi futuri che possono venire intrapresi riguardo questo progetto. L'obiettivo è quello di provare nuove tecniche per capire se sia possibile aumentare ulteriormente le prestazioni dei modelli sviluppati.

Capitolo 2

Strumenti di Apprendimento Automatico

In questo capitolo vengono descritti i fondamenti teorici sui quali si basano le tecniche utilizzate per la costruzione e la validazione dei modelli realizzati. Inizialmente viene presentato e definito il problema teorico della classificazione, nella Sezione 2.2 sono invece descritte le caratteristiche degli alberi decisionali e dei PET. In questa sezione si parla dell'induzione di un albero decisionale e dei problemi che possono sorgere con tale operazione. Inoltre viene descritta la seconda tecnica utilizzata per lo sviluppo del modello finale: SVM. Anche in questo caso viene trattata la fase di induzione, viene inoltre accennato come poter ottenere delle stime di probabilità a partire dall'output di SVM. È presente un'analisi della tecnica applicata assieme a SVM: Random Subspace. In 2.4 viene accennata la tecnica della Regressione Logistica utilizzata nel lavoro del 1998. Infine sono trattate due situazioni tipiche del Data Mining: Missing Values e Dataset sbilanciati; questi argomenti sono presentati a livello teorico, inoltre sono disponibili delle soluzioni che permettono di risolvere tali problematiche.

2.1 Classificazione

Con il termine classificazione si intende l'operazione di assegnare gli oggetti ad alcune categorie predefinite, si tratta di un problema diffuso che riguarda molte applicazioni diverse. Tra gli esempi che si possono citare c'è l'identificazione delle email di spam sulla base dell'Oggetto e del contenuto, la categorizzazione delle cellule come benigne o maligne sulla base dei risultati di una MRI oppure la classificazione delle galassie secondo la loro forma.

Tipicamente l'input per un problema di classificazione è rappresentato da un insieme di dati chiamato *training set*, l'obiettivo è quello di creare una descrizione generale che permetta di catalogare nuovi dati non ancora visti. Il training set può essere descritto in una varietà di linguaggi, solitamente è descritto come un *bag instance* che appartiene ad un certo *bag schema*. (per ulteriori approfondimenti sull'argomento si veda [2]) Un bag instance è un insieme di *tuple* (anche note come record, istanze, o righe) che può contenere duplicati. Ogni tupla è descritta da un vettore di valori associati agli attributi. Un bag schema invece fornisce la descrizione degli attributi e dei loro domini, esso e definito come $\mathcal{A} \cup C$. In questa definizione $\mathcal{A} = \{A_1, A_2, A_3, \ldots, A_m\}$ indica l'insieme degli attributi di input, mentre C è l'insieme che contiene le classi (categorie), a volte viene anche indicato come attributo *target*.

Gli attributi (alcune volte chiamati variabili o feature) sono solitamente di due tipi: nominali oppure numerici, una descrizione completa è data in Tabella 2.1. Quando l'attributo A_i è nominale allora è abitudine definire con $dom(A_i) = \{v_1, v_2, v_3, \ldots, v_{|dom(A_i)|}\}$ il suo dominio di valori dove $|dom(A_i)|$ rappresenta la sua cardinalità finita. In un modo simile il $dom(C) = \Gamma = \{\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_{|dom(A_i)|}\}$ rappresenta il dominio della classe che deve essere necessariamente finito. Gli attributi numerici invece hanno cardinalità infinita.

Lo spazio delle istanze (l'insieme di tutti i possibili esempi) è definito come il prodotto cartesiano tra i domini di tutti gli attributi di input $X = dom(A_1) \times dom(A_2) \times \ldots dom(A_m)$. Lo spazio Universale delle istanze U è invece definito come il prodotto cartesiano tra tutti i domini degli attributi di input e il dominio della classe: $U = X \times \Gamma$.

Formalmente il training set T può essere definito come $T = \{t_1, t_2, \ldots, t_n\}$ dove ciascun record t_q è identificabile come $t_q = \langle x_q, \gamma_q \rangle$ con $x_q \in X$ e $\gamma_q \in \Gamma$.

Definizione 1. *Si definisce classificazione il task di apprendere una funzione target f che mappa ciascun insieme di attributi $\mathcal{A}$ in una delle classi predefinite dell'insieme Γ.*

La funzione target è anche nota in maniera informale come *classification model*.

Formalmente il problema della classificazione è definito con:

INPUT: *Training Set* T formato da n record $T = \{t_1, t_2, \ldots, t_n\}$

- Ogni record t_i ha
  - m attributi (Feature) $t_i(A_1), t_i(A_2), \ldots, t_i(A_m)$
  - Una classe $t_i(C)$
  - $t_i(A_j) \in dom(A_j)$ e $t_i(C) \in \Gamma$

OUTPUT: *Classification Model (*Target Function) f
f viene utilizzata per assegnare una classe ad un record basandosi sulle sue feature

$f : dom(A_1) \times dom(A_2) \times \ldots \times dom(A_m) \rightarrow \Gamma$

Un classificatore permette di risolvere il problema appena definito, esso utilizza un algoritmo di learning $\mathcal{L}$ per poter determinare il modello che meglio identifica la relazione che c'è tra gli attributi dei record e le varie classi. Una volta realizzato il modello è prevista una fase chiamata *validazione* nella quale si va a valutarne la qualità. Solitamente la validazione viene fatta utilizzando un nuovo dataset (*test set*) che ha un bag schema uguale a quello del training set. La fase di validazione prevede che si vadano a confrontare le classi dei record predette dal modello con quelle reali, è perciò necessario che queste siano note a priori.

	Attributi Nominali (Categorici)
Nominali	Dominio di solito discreto sui cui valori non è definibile un ordinamento significativo Operazioni ammissibili: $\{=; \neq\}$ Esempi: Codice Fiscale, colori
Ordinali	Dominio di solito discreto sui cui valori è definito un ordinamento totale e significativo Operazioni ammissibili: $\{=; \neq, \leq, <, >, \geq\}$ Esempi: Voti, Numeri Civici
	Attributi Numerici (Quantitativi)
Discreti	Dominio numerico e discreto Operazioni ammissibili: $\{=; \neq, \leq, <, >, \geq, +, -\}$ Esempi: Età
Continui	Dominio numerico e continuo Operazioni ammissibili: $\{=; \neq, \leq, <, >, \geq, +, -, \div, \times\}$ Esempi: Temperatura, Peso

Tabella 2.1: Tipologia di attributi

2.2 Alberi di Decisione e PET

I decision tree sono una delle tecniche di knowledge discovery più utilizzate. Si tratta di un metodo per approssimare funzioni obiettivo a valori discreti nelle quali la funzione obiettivo è rappresentata attraverso un albero decisionale. Esistono diverse implementazioni di algoritmi di induzione per i decision tree: C4.5 [3], ID3 [4], CART [5] (Classification and Regression Tree), etc, tra questi il più noto ed utilizzato è sicuramente C4.5 sviluppato da Quinlan. Tutti gli argomenti presentati possono essere approfonditi in [6].

2.2.1 Definizione

Definizione 2. *Un albero decisionale o decision tree per un training set T è un albero in cui ogni nodo interno è associato a un test su uno degli attributi e gli archi verso i figli sono etichettati con i risultati distinti del test. Ogni foglia u è associata ad un sottoinsieme di record $T_u \subseteq T$ ed è etichettata con la classe di maggioranza dei record di T_u, inoltre i valori degli attributi dei record associati alla foglia u sono coerenti con i risultati dei test incontrati nel cammino dalla radice alla foglia u. Gli insiemi di record coperti dalle foglie creano una partizione del training set T.*

Questa tipologia di classificatori presenta diversi vantaggi:

- Sono facilmente interpretabili
- Offrono una buona accuratezza in molti problemi di classificazione
- Sono robusti rispetto al rumore e alla ridondanza di attributi
- Vengono costruiti in maniera efficiente.

2.2.2 Tipi di test

Come detto nella definizione, a ciascun nodo interno dell'albero sono associati dei test sugli attributi. Quando gli elementi del training set vengono suddivisi sulla base del risultato di un test si parla di *split*. I test variano a seconda della tipologia di dato sui quali vengono effettuati; è possibile identificare due diversi tipi a seconda che l'attributo sia nominale oppure numerico.

- **Attributo Nominale**
  - Binario: esiste un solo split possibile, i risultati del test sono due ed i record vengono suddivisi a seconda del gruppo a cui $t[A]$ appartiene (e.g $SEX = \{M,\ F\}$)
  - k valori: il dominio dell'attributo ha cardinalità k
    (e.g $Giorni = \{L, Ma, Me, G, V, S, D\}$). Considerando uno split binario si suddividono i valori dell'attributo in due sottoinsiemi disgiunti: Φ e $\bar{\Phi}$ tali che si suddividono i record t con $t[A] \in \Phi$ e con $t[A] \notin \Phi$. In totale è possibile creare $\frac{1}{2} \cdot (2^k - 2)$ sottoinsiemi. Talvolta si possono usare test con più di due risultati partizionando il dominio dell'attributo in intervalli o insiemi di valori

- **Attributo Numerico o Categorico Ordinale**

  – Lo split suddivide i record t con $t[A] \leq x$ da quelli con $t[A] > x$ dove x appartiene al dominio dell'attributo A. Nel caso il test abbia più uscite si definisce un ordinamento tra gli x_i del dominio tale che: $t[A] < x_1$, $x_1 \leq t[A] < x_2$, $x_2 \leq t[A] < x_3$, . . ., $t[A] < x_k$

2.2.3 Algoritmo di Induzione

In linea di principio è possibile costruire molti alberi di decisione a partire da un training set T e da un insieme di attributi, chiaramente alcuni di questi sono più accurati di altri: trovare l'albero ottimo è computazionalmente dispendioso a causa della dimensione esponenziale dello spazio delle soluzioni. Tuttavia sono stati sviluppati algoritmi efficienti in grado di costruire degli alberi decisionali accurati, seppur sub-ottimali, in un tempo computazionale ragionevole. Gran parte di questi algoritmi si basano su una strategia di tipo greedy[1] , costruiscono l'albero compiendo delle decisioni locali ottime riguardo alla scelta dell'attributo da utilizzare nei test per la partizione dei dati. Come già detto esistono diversi algoritmi che permettono di indurre un albero decisionale: *ID3, C4.5, CART, etc.* tutti hanno però in comune il fatto di essere basati su un algoritmo più semplice noto come *Hunt's Algorithm*. Seguendo un approccio di tipo *Divide & Conquer* esso mira ad ottenere sottoinsiemi di record puri, ovvero appartenenti alla stessa classe, partizionando ricorsivamente i record. Esistono due step che sono ripetuti, fino a che l'albero non è completamente costruito:

1. Esamina i dati a disposizione e trova il miglior attributo da utilizzare nel test associato al nodo

2. Suddividi i record sulla base dei risultati del test

3. Riepeti ricorsivamente su ogni nodo figlio scegliendo altri attributi

Viene ora fornito lo pseudocodice dell'algoritmo di Hunt. Sono necessari tre parametri di input: il training set E, l'insieme degli attributi A e l'insieme delle classi C.

[1]Un algoritmo greedy è un algoritmo che cerca di ottenere una soluzione ottima da un punto di vista globale attraverso la scelta della soluzione più golosa (ottima) ad ogni passo locale.

Algoritmo 2.1 Hunt's algorithm

```
1: procedure TreeGrowth(E, A, C)
2:     if StoppingCondition(E, A, C) then
3:         Crea una foglia leaf associata ad E
4:         leaf.label← Classify(E, A, C)
5:     else
6:         Crea un nodo root
7:         root.test_cond ←FindBestSplit (E, A, C)
8:         {V ← v | v possibile risultato del test root.test_cond }
9:         for each v ∈ V do
10:            E_v ← e ∈ E : root.test_cond(e) = v
11:            child ← TreeGrowth(E_v, A, C)
12:            Aggiungi child come foglia di root con un arco etichettato v
13:        end for
14:    end if
15:    return root
16: end procedure
```

Come si può notare esistono tre funzioni che vengono richiamate all'interno dell'algoritmo:

- **FindBestSplit***(E, A, C)*: cerca di determinare quale sia lo split migliore dato il training set *E*. Perché questo sia possibile è necessario identificare delle metriche che permettano di confrontare la bontà dei vari split. L'idea di base è quella di determinare la purezza dell'insieme E prima dello split e quello della partizione indotta dallo split stesso. Viene selezionato il test sull'attributo che permette di massimizzare la purezza. I criteri di valutazione per la purezza vengono definiti nella Sezione 2.2.4.

- **StoppingCondition***(E, A, C)*: permette di determinare quando deve essere fermato il processo di split del training set per creare le foglie dell'albero. Solitamente un modo utilizzato è quello di fermarsi quando la taglia del training set E arriva sotto una certa soglia.

- **Classify***(E, A, C)*: realizza la classificazione dell'insieme di record *E*. Di solito la classe viene assegnata valutando quella che massimizza:

$$\arg\max_i\{p_i : 0 \leq i \leq c-1\} \text{ dove } p_i = \frac{|t \in E : t[C] = \gamma_i|}{|E|}$$

2.2.4 Bontà di uno Split

La funzione cardine per la costruzione di un albero secondo l'algoritmo di Hunt è quella che permette di selezionare il miglior attributo da associare a ciascun nodo interno. Come accennato prima per determinare quale sia il miglior split è necessario identificare un metodo di valutazione, solitamente questo si basa sulla misurazione d'impurità dei nodi figli rispetto a quella del padre. La purezza di uno split è legata alla distribuzione delle classi a cui appartengono i record del nodo: più la distribuzione è uniforme e minore è la purezza del nodo. Considerando uno split che suddivide il training set E in k sottoinsiemi tali che $\bigcup_i E_i = E$, allora l'obiettivo è fare in modo che globalmente i vari E_i siano più puri dell'insieme E. In letteratura esistono diverse misure di impurità, tutte basate sulla distribuzione di probabilità delle varie classi.

Si definisce p_i la frazione dei record di E di classe γ_i dove $\Gamma = \{\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_{c-1}\}$

$$p_i = \frac{|t \in E \,:\, t\,[C] = \gamma_i|}{|E|} \,\mathrm{con}\, 0 \leq i \leq c-1$$

Vale:

$$p_i \in [0,1]\, \forall i \text{ e } \sum_{\mathrm{i=0}}^{\mathrm{c-1}} \mathrm{p_i} = 1$$

Le misure di impurità più note sono *Gini Index*, *Entropy*, *Classification Error*. L'algoritmo di induzione C4.5 utilizza l'Entropia H per determinare il miglior split:

$$H(E) = -\sum_{i=0}^{c-1} p_i \log_2 p_i$$

L'Entropia assume il valore 0 per distribuzioni empiriche di probabilità delle classi di E del tipo

$$p_i = \begin{cases} 1 & i = i^* \\ 0 & i \neq i^* \end{cases}$$

Quando le distribuzioni di probabilità sono uniformi, ovvero $p_i = {}^1\!/_c\, \forall i$ le varie metriche assumono il valore massimo: $H(E) = \log_2 c$.

Note le possibili misure di impurità queste possono venire utilizzate per capire quante più puri siano i nodi figli rispetto al padre, tanto maggiore è la differenza e tanto migliore sarà la condizione di test usata nello split.

Il *gain*, ovvero il criterio che può venire usato per determinare la bontà di uno split è:

Definizione 3. *Il Gain per uno split* $E \to E_1, E_2, \ldots, E_k$ *è definito come:*

$$\triangle = I(E) - \sum_{j-1}^{k} \frac{|E_j|}{|E|} \cdot I(E_j)$$

dove *I(E)* rappresenta una generica misura di impurità. Qualora la misura *I(E)* sia l'*Entropia* allora si parla di *Information Gain:* $\triangle_{info}$

Le misure di impurità definite tendono a selezionare gli split che prevedono molte uscite e suddividono il training set E in molte partizioni. Così facendo c'è il rischio che si costruisca un modello "su misura" dei dati a disposizione che però non funziona altrettanto bene con nuovi record, in questo caso si dice che il modello non generalizza (oppure si parla di *overfitting*). Per ovviare a questo problema si può procedere con due soluzioni: evitare i test che prevedano uscite multiple, obbligando la presenza di soli split binari, oppure penalizzare la presenza di uno split multiplo in fase di costruzione dell'albero. Nel primo caso si alleggerisce il carico computazionale della costruzione dell'albero visto che si generano solo due sottoinsiemi complementari per ogni possibile split. L'algoritmo C4.5 prevede invece una soluzione del secondo tipo, anziché utilizzare il Gain descritto nella Definizione 3 si fa ricorso al *Gain Ratio*:

Definizione 4. *Il Gain Ratio per uno split* $E \to E_1, E_2, \ldots, E_k$ *è definito come:*

$$GainRatio = \frac{\triangle_{info}}{SplitInfo}$$

$SplitInfo = -\sum_{j-1}^{k} \frac{|E_j|}{|E|} \cdot I(E_j)$ dove k rappresenta il numero totale di split. Il valore massimo lo si ottiene quando tutte le partizioni hanno la stessa taglia cioè $|E_j| = {}^{|E|}/_{k} \forall j$ e risulta essere pari a $\log_2 k$. All'aumentare del valore di k si avrà un aumento dello *SplitInfo* e conseguentemente una diminuzione del *GainRatio*. In questo modo si vanno a penalizzare maggiormente i test con uscite multiple rispetto a quelli binari evitando il problema citato sopra.

2.2.5 Overfitting

Molto spesso durante la costruzione e lo sviluppo di un modello si può incorrere in due problemi ben noti: *underfitting* e *overfitting*. Il primo sorge quando il modello creato è troppo semplice (nel caso di un albero decisionale quando vi sono pochi nodi) e non riesce a classificare bene il training set, ne tantomeno il test set. Al crescere della taglia dell'albero diminuisce il numero di errori compiuti sul training e sul test set. Non è però

consigliabile spingersi troppo oltre con l'obiettivo di realizzare un albero estremamente complesso, altrimenti si va incontro al secondo problema chiamato overfitting. In questo caso il modello si adatta estremamente bene al training set, portando a training error molti bassi, ma non riesce a generalizzare, ovvero non riesce a classificare altrettanto bene i record che appartengono al test set. Ad esempio è possibile sviluppare i nodi foglia di un albero fino a quando questi non si adattano in maniera perfetta al training set ottenendo così un training error pari a 0, in questo caso però il generalization error può diventare elevato se si tiene conto che possono essere realizzati dei nodi che si adattano a record estremamente rumorosi.

Possono essere individuate tre cause legate all'overfitting:

- Presenza di rumore tra i record del training set E
- Training set limitato
- Multiple Hyphothesis Testing*:* visto che si usano algoritmi di tipo *greedy* se aumenta lo spazio delle soluzioni aumenta anche la possibilità di imbattersi in un ottimo locale anzichè in uno globale. Questo accade quando il numero degli attributi è elevato oppure quando gli split possibili sono molti e può venirne scelto uno la cui purezza non è ottima.

2.2.6 Pruning

Con gli alberi decisionali è possibile cercare di mitigare il fenomeno dell'overfitting utilizzando una tecnica chiamata *pruning* (potatura). Esistono due tecniche di pruning: *pre-pruning* e *post-pruning*. La prima prevede che si fermi la costruzione dell'albero prima che esso si adatti troppo al training set. Nell'algoritmo di Hunt questa opzione è rappresentata dalla funzione **StoppingCondition***(E, A, C)*, i parametri per fermare la costruzione possono essere:

1. *Gain* ottenuto inferiore ad una certa soglia, non ha più senso continuare a creare split se non si ottengono sostanziali vantaggi
2. Riduzione del generalization error stimato inferiore ad una determinata soglia, vale lo stesso ragionamento fatto per il *Gain*
3. Numero di record di E inferiore ad una determinata soglia, si impone un numero minimo di record che ciascuna foglia deve contenere.

Il *post-pruning* prevede invece un approccio di tipo bottom-up che viene eseguito una volta che la costruzione del modello è stata ultimata, l'obiettivo è quello di eliminare le parti superflue. Una tecnica utilizzata è quella del *subtree-replace* nella quale si cerca di

eliminare interi sottoalberi sostituendoli con delle foglie e riducendo il numero di test che si effettuano nel path da radice a foglia. La nuova foglia raccoglierà tutti i record che erano presenti in quelle dei sottoalberi sostituti [7]. Si procede dalle foglie del modello completo avanzando verso la radice.

L'algoritmo C4.5 utilizza invece un pruning che prende il nome di *error-based-pruning*, i vari nodi vengono potati sulla base di una stima pessimistica del generalization error effettuata sul training set. Ipotizzando di avere un nodo foglia che contiene N record del training set dei quali k sono classificati in maniera errata, allora k/N può essere visto come una stima del contributo della foglia al generalization error. Considerando l'estremo superiore dell'intervallo di confidenza di questa stima l'algoritmo inferisce quello che potrebbe essere il generalization error in un caso pessimo. Così facendo si va a sovrastimare il contributo di ciascuna foglia al generalization error rispetto al rapporto k/N . L'ampiezza dell'intervallo viene gestita impostando un livello di confidenza α. Qualora la stima del generalization error migliori sostituendo il sotto-albero con una foglia allora avviene la potatura.

2.2.7 PET: Probability Estimation Tree

Molto spesso etichettare i record con la classe di appartenenza non è sufficiente, per aumentare le informazioni che il modello fornisce si può associare a ciascuna istanza la probabilità di appartenere ad una determinata classe. In ambito medico questo è molto importante poiché semplicemente definire un paziente "malato" è riduttivo e spesso può portare ad errori; si preferisce invece fornire una percentuale che indica l'appartenenza del paziente alla classe "malato" in modo che il medico possa effettuare le proprie conclusioni. Avendo a disposizione la distribuzione di probabilità rispetto alle varie classi la fase di classificazione avviene definendo un valore soglia e discriminando le istanze sulla base di esso. Se si associa una distribuzione di probabilità alle foglie di un albero decisionale si parla di *PET: Probability Estimation Tree*. Essi hanno le stesse caratteristiche dei Decision Tree descritti in 2.2.1 ed inoltre permettono una classificazione basata su probability-ranking [8].

Definizione 5. *Un Probability Estimation Tree (PET) per un training set T è un albero decisionale dove a ciascun nodo u è associata la distribuzione di probabilità relativa alle classi dei record T_u contenuti in esso.*

Dalla definizione si nota come i valori di probabilità siano presenti anche nei nodi interni e non solamente nelle foglie. Ai fini della classificazione dei record le sole probabilità che vengono usate sono quelle delle foglie, i valori presenti nei nodi interni servono a titolo informativo.

Un esempio di PET è mostrato in Figura 2.1, si tratta di un problema medico di classificazione binaria che stima la possibilità che un paziente ha di guarire rispetto ad una malattia. Come si nota esso è strutturalmente identico ad un Decision Tree (in questo caso al posto dei nodi sono stati inseriti dei grafici a torta che permettono immediatamente di capire le distribuzioni di probabilità, al colore verde è associata la possibilità di guarigione). Dall'analisi di un PET si possono trarre più conclusioni rispetto ad un normale albero, ad esempio si vede che i maschi hanno minore probabilità di guarire rispetto alle femmine, oppure che le femmine più giovani hanno una prospettiva di guarigione più elevata.

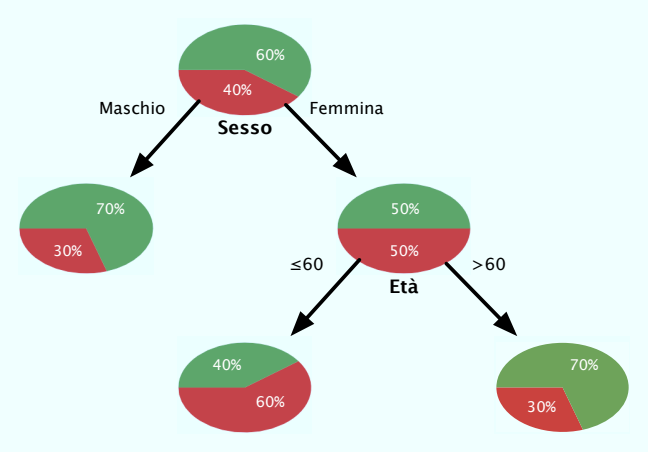


Figura 2.1: Esempio di PET

Induzione di PET

Un decision tree può essere facilmente modificato in modo da ottenere un PET utilizzando la frequenza assoluta dei record che si trovano nelle varie foglie. Si consideri un istanza t che viene mappata su una foglia a cui è associato un insieme di record E, allora la probabilità che essa appartenga alla classe γ è:

$$P_M(t\,[C] = \gamma) = \frac{|\{t \in E \,:\, r\,[C] = \gamma\}|}{|E|}$$

Questo tipo di approccio è stato studiato per primo da Provost in [9] e viene indicato come metodo di *massima verosimiglianza*. L'algoritmo di induzione per i PET può essere ricavato in maniera semplice dall'Algoritmo 2.1 andando a sostituire la funzione *Classify(E, A, C)* con *EstimateProbabilities(E, A, C)* che utilizza al suo interno la stima della probabilità citata sopra. Tale stima presenta però dei limiti dovuti al fatto che si va calcolare la probabilità esclusivamente utilizzando l'insieme E, perciò quando la sua cardinalità è molto bassa si ottengono dei valori poco precisi.

Si supponga un insieme di record E associati ad una foglia u e definiti su uno schema di attributi $\mathcal{A} \cup C$. Si ipotizzi un problema binario ($C = 2$) e si consideri che nella foglia sono presenti solamente 5 elementi ($|E| = 5$). Ogni elemento del nodo appartiene alla stessa classe γ_0, la stima della probabilità secondo il metodo della massima verosimiglianza porta a:

$$P_M(t\,[C] = \gamma_0) = \frac{|\{t \in E \,:\, r\,[C] = \gamma_0\}|}{|E|} = \frac{5}{5} = 1$$

Questo vorrebbe dire che tutti i record che cadono in quella foglia hanno una probabilità di appartenere alla classe γ_0 del 100%, ciò sembra una stima poco veritiera. Come si vede, per foglie "pure", si hanno estremi valori di probabilità. Per cercare di mitigare il problema si ricorre a delle tecniche di *smoothing*, quella più semplice e nota è la *Correzione di Laplace*. Ciò che viene fatto è incorporare una probabilità *a priori*[2] pari a $1/c$ in ciascuna classe , si noti che con zero esempi ciascuna classe ha pari probabilità: $1/c$. La correzione di Laplace porta a:

$$P_M(t\,[C] = \gamma) = \frac{|\{t \in E \,:\, r\,[C] = \gamma\}| + 1}{|E| + c}$$

dove c indica il numero delle classi. Nell'esempio di prima dunque:

$$P_M(t\,[C] = \gamma_0) = \frac{|\{t \in E \,:\, r\,[C] = \gamma_0\} + 1|}{|E| + c} = \frac{5+1}{5+2} \approx 0.86$$

La probablità calcolata sembra dunque più realistica. Molti esperimenti, [9] e [8], hanno dimostrato che le tecniche di smoothing portano ad un miglioramento delle performance.

Analizzando quanto detto fino ad ora potrebbe sembrare che i PET non siano uno strumento adatto per la stima di probabilità come potrebbe esserlo uno numerico che esegue la stima rispetto alle classi fornendo un output continuo (di tipo smooth) in un intervallo [0,1]. In realtà invece come confermato da Provost i PET possono essere ottimi stimatori di probabilità. Si prendano le due tipologie di attributo: nominale (categorico) e numerico. Nel primo caso l'albero può stimare la distribuzione di probabilità in modo preciso andando a fornire uno split, e dividendo il training set, sugli stessi valori dell'attributo categorico. Nel caso di un attributo nominale continuo un PET sufficientemente ampio può stimare ogni funzione di probabilità per le classi con una precisione arbitraria. In Figura 2.2a è mostrata la distribuzione di probabilità di una classe rispetto ad un attributo continuo. A seconda dei valori assunti dall'attributo A si hanno probabilità diverse per la classe in questione. Si consideri ad esempio uno split del tipo $t\,[A] \in (-\infty, -1.5]\,;\, t\,[A] \in (-1.5, 1.5]\,;\, t\,[A] \in (1.5, +\infty]$, un PET può stimare la probabilità, con un certo livello di dettaglio. Per fare ciò può "idealmente" tracciare dei

[2] *Probabilità a priori: probabilità che viene stimata prima che l'evento E si verifichi*

rettangoli posizionati sui segmenti identificati dai valori di split ed analizzarne l'altezza, in questo modo ottiene una stima della probabilità. È intuitivo capire che utilizzando uno split su più valori (aumentando quindi la dimensione del PET) si ottiene una stima della probabilità migliore (linea verde): tanti più rettangoli si creano e migliore sarà l'approssimazione della curva. Nei limiti ci si può spingere a predizioni di probabilità ottime utilizzando PET molto complessi.

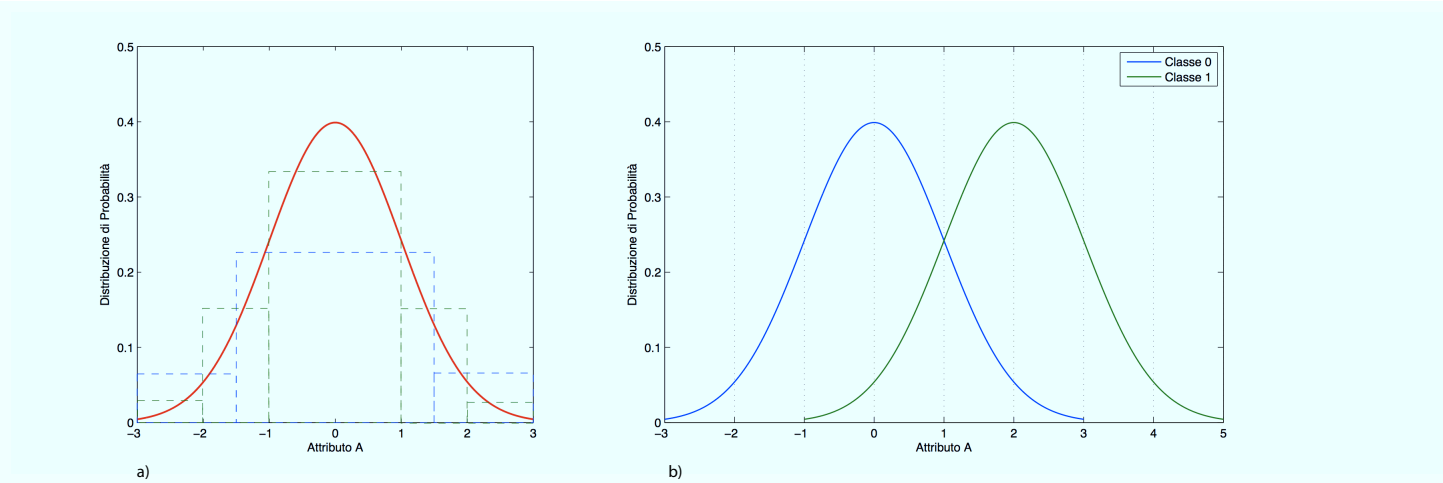


Figura 2.2: a) Distribuzione di probabilità per l'attributo A b) Distribuzione di probabilità per l'attributo A in un problema binario

Ora rimane però da comprendere come mai i PET prodotti da algoritmi standard non portino spesso a buone stime di probabilità. Provost nel suo lavoro afferma che ciò non è dovuto alla struttura degli alberi, come appena dimostrato, ma piuttosto al loro algoritmo di costruzione. Storicamente si è sempre valutato un decision tree rispetto a due criteri: accuratezza e dimensione, si cerca sempre di trovare un buon compromesso rispetto a questi parametri. Si consideri però un problema binario univariato, in Figura 2.2b vi sono le distribuzioni di probabilità delle due classi rispetto all'unico attributo A. Si tratta di un'estensione del caso mostrato in Figura 2.2a considerando contemporaneamente le due classi. Scegliendo un solo split ad $x = 1$ si ottiene un albero con la massima accuratezza e la minima taglia. Con tale split si crea un albero che separa le classi bene come qualsiasi altro realizzabile e si raggiunge anche la dimensione più piccola possibile. Così facendo però la stima della probabilità dell'albero non è precisa: tutti i punti che si trovano sullo stesso lato dello split hanno la medesima probabilità che corrisponde alla proporzione della classe che cade in quello stesso lato. Dato che le tecniche di pruning, descritte precedentemente, vengono utilizzate per cercare di ridurre la dimensione dell'albero, e

trovare un'elevata accuratezza, è imputabile a loro una riduzione della precisione nelle stima della probabilità.

2.3 SVM

SVM (Support Vector Machine) è una tecnica di apprendimento supervisionato che permette di risolvere il problema della classificazione di pattern. Essa ha radici nella teoria di learning statistico, è stata sviluppata negli anni '90 da *Vladimir Vapnik* presso i laboratori Bell AT&T [10]. Una SVM è un classificatore binario che apprende il confine tra esempi appartenenti a due classi diverse. Proiettando gli esempi in uno spazio multidimensionale (di dimensione opportunamente scelta) si cerca il miglior iperpiano di separazione dei campioni. L'iperpiano scelto deve massimizzare la sua distanza (il "margine") dagli esempi di training più vicini.

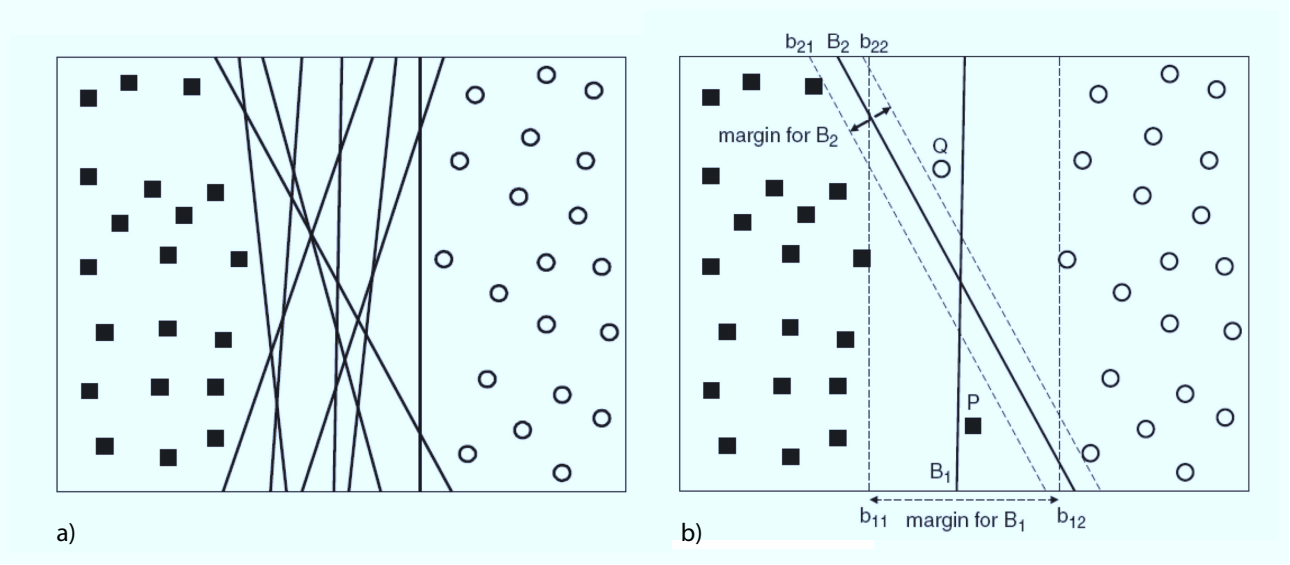


Figura 2.3: a) Possibili iperpiani di separazione per un dataset linearmente separabile b) Margine di un decision boundary

Per illustrare in maniera corretta il funzionamento delle SVM è necessario introdurre il concetto di *Maximum Margin Hyperplane*. Si consideri la Figura 2.3a che mostra un insieme di dati appartenenti a due classi diverse. Il dataset è *linearmente separabile* ovvero si può trovare un iperpiano tale che: gli esempi di una classe risiedono su un suo lato mentre quelli dell'altra classe su quello opposto. Dalla Figura si vede che esistono infiniti iperpiani che possono dividere i due insiemi, anche se tutti hanno un training error nullo non è detto

che si comporteranno bene anche con esempi non visti. La scelta deve essere fatta in modo da ottenere il miglior risultato su un nuovo test set. In Figura 2.3b sono mostrati due iperpiani diversi, a ciascun decision boundary B_i è associata una coppia di iperpiani b_{i1} e b_{i2}, la distanza tra di essi è identificata come *margine* del classificatore. Gli iperpiani con margine maggiore tendono ad avere un miglior generalization error rispetto a quelli con margine limitato. Intuitivamente se il margine è molto sottile ogni piccola perturbazione sugli iperpiani può avere forti ripercussioni sulla classificazione, inoltre classificatori con un margine minore sono più soggetti all'overfitting.

2.3.1 Linear SVM: Caso Separabile

Si consideri un problema di classificazione binaria con N esempi di training. Ogni esempio è una tupla del tipo (x_i, y_i) dove $x_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{id})^T$ è un vettore che identifica i valori degli attributi del record. Convenzionalmente y_i rappresenta la classe $C = \{+1, -1\}$. Il confine di classificazione (*decision boundary*) di un classificatore lineare può essere scritto come:

$$\mathbf{w} \cdot x + b = 0$$

dove **w** e b sono i parametri del modello.

Considerando ancora la Figura 2.3 ogni istanza che appartiene all'iperpiano deve soddisfare l'equazione appena scritta. Ad esempio presi due punti x_a e x_b:

$$\mathbf{w} \cdot x_a + b = 0$$

$$\mathbf{w} \cdot x_b + b = 0$$

Sottraendo le due equazioni si ottiene

$$\mathbf{w} \cdot (x_a - x_b) = 0$$

dove $x_b - x_a$ è un vettore parallelo all'iperpiano e diretto da x_a a x_b. Dato che il prodotto vettoriale è zero la direzione per **w** deve essere perpendicolare all'iperpiano.

Per ogni quadrato che si trova sopra il decision boundary si può mostrare che:

$$\mathbf{w} \cdot x_s + b = k$$

dove $k > 0$, allo stesso modo per ogni cerchio che si trova sotto:

$$\mathbf{w} \cdot x_q + b = k'$$

dove $k' < 0$. Se si etichettano i quadrati con la classe $+1$ e i cerchi con -1 allora si può predire la classe y per qualsiasi esempio di test z nel seguente modo:

$$y = \begin{cases} 1, & se\ \mathbf{w} \cdot \mathbf{z} + b > 0 \\ -1, & se\ \mathbf{w} \cdot \mathbf{z} + b < 0 \end{cases}$$

È possibile riscalare i parametri $\mathbf{w}$ e b dell'iperpiano in modo tale che i due iperpiani b_{i1} e b_{i2} possano essere così definiti:

$$b_{i1} : \mathbf{w} \cdot \mathbf{x} + b = 1$$

$$b_{i2} : \mathbf{w} \cdot \mathbf{x} + b = -1$$

Come detto, il margine del decision boundary è dato dalla distanza di questi due iperpiani. Sia x_1 un punto che si trova su b_{i1} e x_2 uno che sta su b_{i2} come mostrato in Figura 2.4, allora sostituendo i punti nelle due equazioni sopra il margine può essere calcolato sottraendo la prima con la seconda:

$$\mathbf{w} \cdot (x_1 - x_2) = 2 \qquad \|w\| \times d = 2$$

$$d = \frac{2}{\|w\|} \tag{2.1}$$

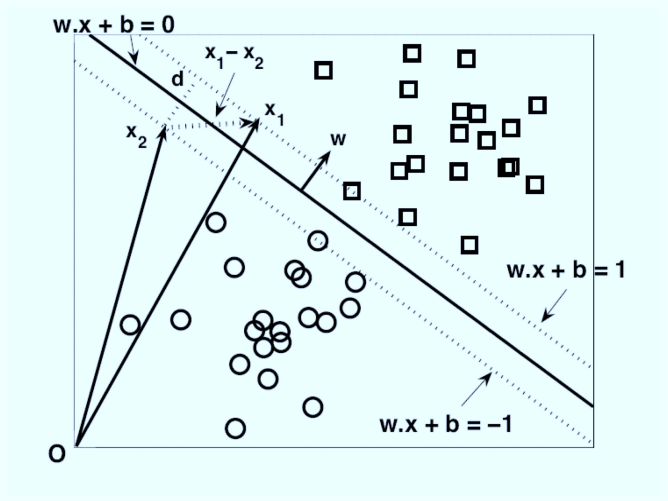


Figura 2.4: Margine per un decision boundary

Learning di un modello SVM

La fase di learning di una SVM implica la stima dei parametri $\mathbf{w}$ e b dell'iperpiano a partire dai record del training set. I parametri devono essere scelti in modo tale che siano garantite due condizioni:

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1 \text{ se } y_i = 1 \tag{2.2}$$

$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 \text{ se } y_i = -1 \tag{2.3}$$

Esse impongono che tutte le istanze della classe $y = 1$ siano collocate al di sopra o sull'iperpiano mentre quelle della classe -1 siano collocate al di sotto o sull'iperpiano stesso. Si possono sommare le due condizioni in una unica:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 \;\; i = 1, 2, 3, \ldots, N \tag{2.4}$$

Oltre a questo bisogna imporre che il margine dell'iperpiano sia massimale. Cercare di massimizzare il margine, definito in 2.1, è equivalente a minimizzare la seguente funzione obiettivo:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} \tag{2.5}$$

Il problema della fase di addestramento di una SVM può essere ora formalizzato come:

Definizione 6. *La fase di learning in SVM è definita dal seguente problema di ottimizzazione vincolata:*

$$\min_w \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{soggetto a } \; y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 \;\; i = 1, 2, 3, \ldots, N$$

La risoluzione del seguente problema di ottimizzazione convessa avviene con il metodo dei *moltiplicatori di Lagrange,* in questo modo si possono determinare i parametri per l'equazione dell'iperpiano e la classificazione di un qualsiasi record z avviene sulla base di:

$$f(z) = sign(\mathbf{w} \cdot \mathbf{z} + b) = sign\left(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x_i} \cdot \mathbf{z} + b\right)$$

Se $f(z) = 1$ allora il record di test è classificato come positivo, altrimenti come negativo.

2.3.2 Linear SVM: Caso non separabile

Non tutti i dataset sono linearmente separabili, si consideri la Figura 2.5a che aggiunge due nuovi esempi rispetto alla Figura 2.3. Anche se B_1 classifica correttamente i due nuovi esempi non è detto che sia migliore di B_2, anche perché P e Q potrebbero essere dovuti al rumore presente nel dataset. La costruzione trattata nella sezione precedente prevede solo che si identifichino degli iperpiani che suddividono i record, in base alle classi di appartenenza, senza commettere errori. L'idea è quella di apprendere un decision boundary che sia tollerabile ad un numero limitato di errori rispetto al training set, si tratta di un metodo noto come approccio *soft margin*. Questa metodologia prevede un tradeoff tra la larghezza del margine ed il numero di errori commessi rispetto al training set.

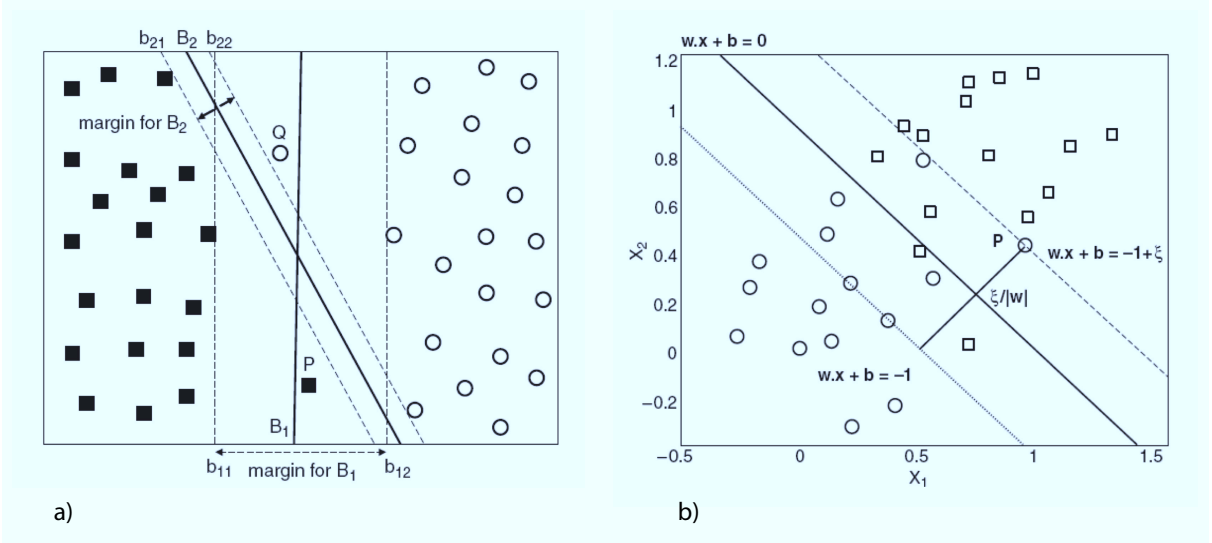


Figura 2.5: a) Decision boundary in un caso non separabile b) Variabile di slack

I vincoli presentati precedentemente vengono rilassati introducendo una variabile *slack* ξ positiva:

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1 - \xi_i \; se \; y_i = 1$$

$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 + \xi_i \; se \; y_i = -1$$

dove $\forall i : \xi_i > 0$.

Per capire il significato della variabile di slack si consideri la Figura 2.5b. Il punto P viola i vincoli definiti in 2.2 e 2.3. Sia $\mathbf{w} \cdot \mathbf{x} + b = -1 + \xi$ una linea parallela all'iperpiano

che passa per P. Si può dimostrare che la distanza tra l'iperpiano e quella linea è $\xi/\|\mathbf{w}\|$, per cui ξ fornisce una stima dell'errore del decision boundary sull'esempio di training P. In linea di principio si può usare la funzione obiettivo del caso seperabile ed imporre le stesse condizioni, ma dato che non vi sono vincoli sul numero di errori che possono essere commessi si può trovare un iperpiano che ha un margine elevato ma commette molti errori. Per evitare questo si modifica la funzione obiettivo in modo da penalizzare gli iperpiani con un elevato valore per la variabile di slack:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C\left(\sum_{i=2}^{N} \xi_i\right)^k \tag{2.6}$$

dove C e k sono parametri specificati dall'utente che rappresentano la penalità nel commettere errori di classificazione. C può essere scelto sulla base delle performance del modello valutate su un validation set, controlla il tradeoff tra errori di training e massimizzazione del margine ($C = \infty$ porta ad una situazione di *hard margin,* abbassando il valore aumentano gli errori sul training set).

2.3.3 Non Linear SVM

Quanto proposto fino ad ora può essere applicato se i dati hanno delle decision boundaries di tipo lineare, qualora si presenti una situazione come quella di Figura 2.6a è necessario applicare una nuova tecnica, che prevede di trasformare i dati dal loro spazio di coordinate originali $\mathbf{x}$ in un nuovo spazio $\Phi(\mathbf{x_i})$ tale che: un iperpiano possa essere usato per separare le istanze di classi diverse nel nuovo spazio. Una volta fatto questo si può applicare la stessa tecnica presentata nella sezione precedente.

L'approccio della trasformazione degli attributi presenta però alcuni problemi di implementazione: non è chiaro che tipo di funzione utilizzare per mappare i dati nel nuovo spazio per ottenere una divisione lineare. Una possibilità è quella di usare uno spazio a dimensione infinita ma questo risulta difficile da trattare. Inoltre anche conoscendo la funzione di mapping risulta difficile lavorare in un spazio con molte dimensioni.

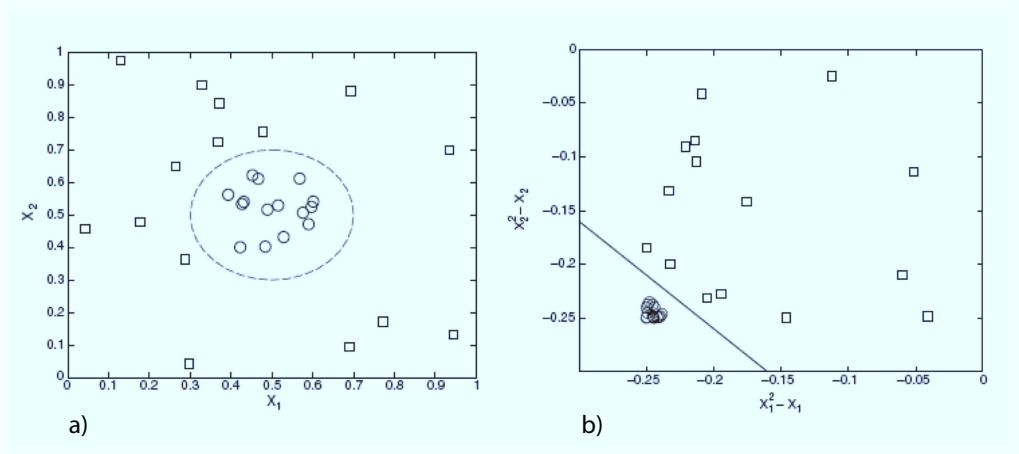


Figura 2.6: a) Decision boundary nello spazio originale b) Decision boundary nello spazio trasformato

Per definire il nuovo problema si utilizza la funzione di mapping Φ che porta al nuovo spazio desiderato:

$$\min_w \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{soggetto a } \; y_i(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i}) + b) \geq 1 \;\; i = 1, 2, 3, \ldots, N$$

In maniera analoga al caso lineare un'istanza di test z viene classificata usando:

$$f(z) = sign(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{z}) + b) = sign\left(\sum_{i=1}^{N} \lambda_i y_i \Phi(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{z}) + b\right)$$

Si noti che, eccetto l'equazione da cui si ricava **w,** tutte le altre utilizzano il prodotto scalare tra una coppia di vettori all'interno dello spazio trasformato: $\Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j})$. Questo può essere poco maneggevole soprattutto quando si hanno molte dimensioni, per risolvere ciò si fa ricorso al *Kernel*. Se esiste una funzione K tale che: $K(\mathbf{x_j}, \mathbf{x_i}) = \Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j})$ allora l'algoritmo di training può utilizzare solo K senza dover conoscere la natura di Φ. La funzione utilizzata per classificare un nuovo record z diviene allora:

$$f(z) = sign\left(\sum_{i=1}^{N} \lambda_i y_i K(\mathbf{x}_i, \mathbf{z}) + b\right)$$

Di seguito si riportano alcune funzioni che rispettano il teorema citato e sono tra le più utilizzate:

- Polinomiale: $K(x, y) = (x \cdot y + 1)^p$
- Gaussiana (Radial Basis Function): $K(x, y) = e^{-\|x-y\|^2\gamma}$
- Tangente Iperbolica: $K(x, y) = \tanh(kx \cdot y - \delta)$

Il buon funzionamento di una SVM dipende dalla selezione del Kernel, dalla scelta dei suoi parametri e dal parametro di soft margin C. Una comune adozione è quella di utilizzare il Kernel *Gaussiano* che ha un solo parametro γ. La migliore combinazione di C e γ è spesso trovata attraverso una *Grid Search* che prevede una scelta esponenziale dei loro valori, ad esempio:$C \in \{2^{-5}, 2^{-3}, \ldots, 2^{13}, 2^{15}\}$; $\gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{1}, 2^{3}\}$. Soltiamente ciascuna combinazione di valori viene controllata utilizzando una Cross Validation, la coppia che garantisce risultati migliori viene poi scelta. Il modello finale è costruito utilizzando l'intero dataset sulla base dei parametri selezionati con la ricerca.

2.3.4 Stima della probabilità con SVM

Si può facilmente intuire che la funzione di decisione di SVM: $f(x) = \mathbf{w} \cdot \Phi(x) + b$ restituisce la distanza, nello spazio delle feature, dell'esempio trasformato $\Phi(x)$ dall'iperpiano di separazione. Perciò a differenza dei PET non è possibile ottenere in output direttamente le probabilità di appartenenza alle relative classi. Assumendo che $P(Y = 1 \mid x)$ sia continua in x, sembra ragionevole pensare che gli esempi che risiedono vicino all'iperpiano abbiano un'elevata probabilità di essere classificati erroneamente rispetto agli esempi che stanno più lontano. Più vicine è l'esempio all'iperpiano e minori sono i cambiamente che deve avere per essere classificato diversamente. Perciò appare conveniente modellare la probabilità condizionata per le classi $P(y \mid x)$ come una funzione del valore della funzione di decisione di SVM: i.e $\hat{P}(Y = 1|x) = \sigma(f(x))$ dove σ è una appropriata funzione di scaling. Per giustificare l'interpretazione di $\hat{P}(Y = 1|x) = \sigma(f(x))$ è ragionevole utilizzare i record del dataset per calibrare la funzione di scaling. Possono essere usati dei record che non sono stati considerati per il training, oppure si può ricorrere ad un approccio simile alla Cross Validation. In entrambi i casi si cerca di ottimizzare σ per minimizzare l'errore tra la probabilità della classe predetta $\sigma(f(x))$ e la probabilità empirica della classe definita dai valori y. Ci sono diverse misure solitamente usate, ad esempio *Cross-Entropy*:

$$CRE = \sum_i y_i \log(z_i) + (1 - y_i) \log(1 - z_i)$$

che è la distanza di *Kullback-Leibler* tra la probabilità predetta e la probabilità empirica della classe, dove $z_i = \sigma(f(x_i))$.

Il problema fondamentale resta quello di determinare quale funzione di scaling utilizzare. Una tra le più note ed usate è stata presentata e motivata da analisi empiriche in [11] da Platt. Si tratta di una funzione di scaling della forma:

$$\sigma_{a,b}(z) = \frac{1}{1 - e^{-az+b}}$$

con $a \geq 0$ in modo da ottenere una funzione monotonica crescente. I parametri *a, b* sono determinati minimizzando l'errore di Cross-Entropy CRE su un test set $(x_i, y_i)_{i=1,\dots n}$ con $z_i = f(x_i)$. Platt nel suo articolo fornisce uno pseudocodice per risolvere tale problema. In questo modo, a partire dagli score SVM, si può calcolare un valore di probabilità per ciascun record del dataset.

2.4 Logistic Regression

Nel lavoro presentato nel 1998 si è utilizzata una tecnica diversa rispetto a quelle presentate finora: la *Logistic Regression*. In statistica la Regressione Logistica è usata per predire la probabilità di occorrenza di un evento binario (dicotomico) utilizzando una funzione logistica. Si tratta di un modello di generalizzazione lineare usato per la regressione binomiale dove sono presenti una o più variabili predittive (indipendenti) che possono essere numeriche o nominali: $X = \{x_1, x_2, x_3, \dots x_k\}$. L'obiettivo è quello di analizzare la relazione tra la probabilità di successo della variabile di output (dipendente) y e le variabili indipendenti. Ad esempio la probabilità che una persona abbia un infarto a partire dai alcuni fattori di rischio quali età, peso, colesterolo, etc.

Per capire la Logistic Regression è necessario spiegare cos'è una logistic function:

$$f(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

Così come la probabilità, la logistic function assume sempre valori compresi tra 0 ed 1. Un esempio grafico della funzione logistica è mostrato in Figura 2.7. L'input è dato dalla variabile z mentre l'output è $f(z)$. Si tratta di una funzione utile perché può prendere in ingresso ciascun valore reale da $+\infty$ a $-\infty$ e mantenere il suo valore di uscita sempre limitato a [0,1]. La funzione $f(z)$ rappresenta la probabilità di un particolare risultato quando in input si hanno le variabili indipendenti $x_1, x_2, \dots x_k$. La variabile z è una misura del contributo totale di tutte le variabili indipendenti usate nel modello, solitamente è definita come:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k$$

dove β_0 è chiamato *intercept* e $\beta_1, \beta_2, \dots \beta_k$ sono i coefficienti della regressione delle variabili $x_1, x_2, \dots x_k$ rispettivamente. L'intercept rappresenta il valore di z quando tutte le variabili x_i sono nulle (probabilità quando non ci sono fattori di rischio), mentre i coefficienti di regressione indicano la grandezza del contributo di quel fattore di rischio.

Se il coefficiente è positivo allora significa che la variabile a cui è associato aumenta la probabilità dell'evento di output, se negativo significa che la diminuisce, un coefficiente con valore vicino a 0 significa che ha una piccola influenza sulla variabile di uscita.

La logistic regression è quindi un modo utile per descrivere la relazione, tra una o più variabili dipendenti ed una variabile di output binaria, espressa come probabilità.

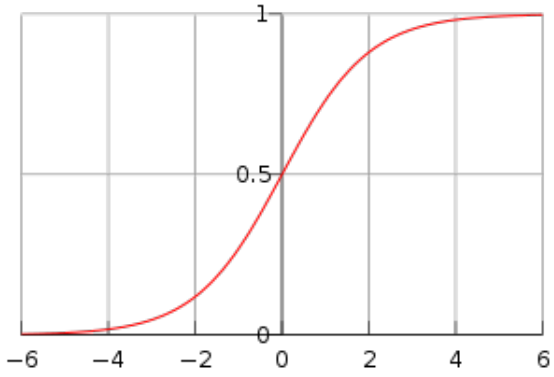


Figura 2.7: Logistic Function

2.5 Random Subspace

A causa dell'elevata dimensionalità dello spazio delle feature può essere difficile per un metodo multivariato (SVM, Decison Tree, Bayesian Network, etc) identificare le variabili rilevanti. Il *Random Subspace* è una tecnica di ensemble learning proposta da Ho [12] che cerca di risolvere questo problema. Sia T il training set formato da n record $T = \{t_1, t_2, \ldots, t_n\}$ nel quale ciascun $t_i = \left(A_1^i, A_2^i, \ldots, A_p^i\right)$ è un vettore di dimensione p. Nel Random Subspace vengono selezionate $r < p$ feature dal dataset T e viene definito un nuovo training set $T^* = \{t_1^*, t_2^*, \ldots, t_n^*\}$ dove ciascun record consiste di un vettore r-dimensionale: $t_i = (A_1^i, A_2^i, \ldots, A_r^i)$ le cui r componenti sono scelte in maniera random a partire dalle p originali. La selezione è la stessa per ogni istanza del dataset. Il classificatore viene quindi allenato utilizzando questo nuovo training set T^* a dimensione minore. Il processo viene ripetuto b volte in modo da ottenere b sottoinsiemi di feature e b modelli, l'output finale del classificatore è dato dalla combinazione dei punteggi dei singoli classificatori.

Algoritmo 2.2 Random Subspace Method

procedure RSM(T, b, r)
  for i = 1 **to** b **do**
    Seleziona r feature dall'insieme iniziale $\mathcal{A} = \{A_1, A_2, \ldots, A_p\}$
    Crea il training set $T^* = \{t_1^*, t_2^*, \ldots, t_n^*\}$ con $t_i = (A_1^i, A_2^i, \ldots, A_r^i)$
    Costruisci il classificatore M_i su T^*
  end for
  Combina gli output degli M_i : e.g $score_{S.R}(t_i) = \frac{1}{b}\sum_{j=1}^{b} score_{M_j}(t_i)$
  return $score_{S.R}$
end procedure

Quando il numero di istanze del training set è relativamente piccolo, se comparato con la dimensione dello spazio delle features, costruire un classificatore su un sottospazio casuale può risolvere il problema del basso numero di esempi. La dimensione del nuovo spazio è inferiore rispetto all'insieme originale mentre il numero di esempi rimane costante, perciò il numero relativo di record di training aumenta. Quando i dati hanno variabili ridondanti la classificazione può essere migliore in un Random Subspace piuttosto che nello spazio di partenza, le prestazioni combinate di più modelli ricavate in questo modo possono aumentare rispetto a quelle di un singolo modello creato sull'insieme originale.

2.6 Metriche di Prestazione

All'interno di uno schema di classificazione la fase di validazione permette di capire quanto bene funziona il modello che è stato sviluppato. Non ha molto senso basarsi sul numero di record del training set classificati correttamente dato che la funzione è stata costruita su di essi, piuttosto si utilizza un test set formato da *dati blind* (mai utilizzati) per analizzare la qualità del lavoro.

Visto che anche gli elementi del training set possono essere classificati in maniera errata si introducono due tipi di errore: *training error* e *generalization error*. Ricordando che U è l'universo di tutti i possibili record e che vale la relazione per cui il training set $E \subset U$ si definisca:

$$M(t) \equiv \text{classe predetta dal modello } M \text{ per il record } t$$

$$t(C) \equiv \text{classe reale del record } t$$

Allora:

- *Training error* con $t \in E$: $M(t) \neq t(C)$
- *Generalization error* con $t \in U \setminus E$: $M(t) \neq t(C)$

Si può dunque calcolare il *training error rate* come:

$$\frac{|\{t \in E \,:\, M(t) \neq t(C)\}|}{|E|}$$

Per quanto riguarda invece il *generalization error rate* esso è solamente stimabile visto che si cerca di descrivere un fenomeno aleatorio quale la percentuale di record classificati erroneamente tra quelli non ancora osservati.

Quello che noi sappiamo è che nel test set il modello ha ottenuto un particolare error rate, ad esempio 25%, ora come possiamo utilizzare questo valore per trarre delle conclusioni riguardo al *vero* error rate rispetto alla popolazione target U? Sicuramente il valore sarà vicino a 25%, ma quanto vicino? Entro il 5%? Entro il 10%? Chiaramente tanto più è grande il test set e tanto più si avrà una stima plausibile. Per avere però una risposta certa bisogna fare qualche ragionamento statistico. Una serie di eventi indipendenti che hanno due possibili risultati, successo o fallimento, è chiamata *Processo di Bernoulli*. L'esempio più classico è il lancio di una moneta in maniera consecutiva (e.g testa=successo e croce=fallimento). Supponiamo che la moneta sia sbilanciata (la probabilità p di avere testa è diversa da 0.5), ma non sappiamo di quanto. Se si lancia la moneta 100 volte e 75 di queste sono testa allora possiamo dire, come per il classificatore sopra, che vi è un error rate pari a 25%. Cosa possiamo concludere rispetto a p? Generalizzando l'esperimento con N lanci e S successi allora l'error rate $e = 1 - S/N$ come può essere legato a p? La risposta a questa domanda è solitamente espressa attraverso un intervallo di confidenza, ovvero p risiede in un certo intervallo di valori con una certa confidenza. Ad esempio se S=750 e N=1000 allora $e = 0.25$. Ma nella realtà quanto siamo vicini al valore di p? Possiamo dire che con una confidenza del 80% p risiede tra 73.2% e 76.7%. Come si arriva a questi valori?

La media e la varianza di un singolo esperimento bernoulliano con probabilità p sono rispettivamente p e $(1-p)\cdot p$. Se si fanno N tentativi in un processo di Bernoulli il *success rate* che ci si aspetta (*success rate* = 1-*error rate* quindi *f*=*1-e*) $f = S/N$ è una variabile aleatoria con media p e varianza ridotta di un fattore N cioè: $\frac{(1-p)\cdot p}{N}$. Per valori elevati di N allora la distribuzione di questa variabile approssima la distribuzione normale (si danno per note queste nozioni statistiche che possono essere riviste in [13]).

La probabilità che una variabile aleatoria X, con media nulla, risieda in un certo intervallo di confidenza $2z$ è:

$$Pr\left[-z \leq X \leq z\right] = c$$

Per le distribuzioni normali i valori di c e i relativi valori di z sono dati in apposite tabelle. Solitamente però vengono date informazioni diverse ovvero:

$$Pr\left[X \geq z\right]$$

Questa rappresenta solamente l'estremità "superiore" della distribuzione. Dato che le distribuzioni normali sono simmetriche allora la probabilità per la parte "inferiore" $Pr\left[X \leq -z\right]$ è la stessa.

Ora l'ultima cosa da fare è quella di trasformare la variabile f in modo da avere media nulla e varianza unitaria. Per fare questo si sottrae la media e si divide per la deviazione standard $\sqrt{p \cdot (1-p)/N}$ ottenendo:

$$Pr\left[-z < \frac{f-p}{\sqrt{p \cdot (1-p)/N}} < z\right] = c$$

Questa è la procedura per determinare l'intervallo di confidenza. Dato un particolare valore per c si utilizzano le apposite tabelle per i relativi valori di z.

L'ultimo step è prevede la risoluzione di una equazione quadratica che porta ad una formula finale per l'intervallo di confidenza:

$$p = \left(f + \frac{z^2}{2N} \pm z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}\right) \Big/ \left(1 + \frac{z^2}{N}\right)$$

Il simbolo $\pm$ porta a due valori per p che rappresentano il limite superiore ed inferiore dell'intervallo. Questa formula può essere utilizzata ora per ottenere i risultati del precedente esempio numerico.

Utilizzando il *generalization error rate* si può definire l'accuracy di un modello come:

$$accuracy = 1 - generalization\ error\ rate$$

Essa rappresenta la frazione di record classificati in maniera corretta rispetto alla popolazione.

Utilizzare la sola accuratezza come parametro di valutazione di un modello porta a delle conclusioni sbagliate in determinati casi. Questo problema è molto evidente nelle situazioni in cui vi è un forte sbilanciamento tra il numero dei record appartenenti alle diverse classi, si parla di *Imbalance Problem* (dataset sbilanciati) e viene trattato nella Sezione 2.8. Si supponga di voler sviluppare un modello che permetta l'identificazione

di transazioni fraudolente in un dataset che contiene transazioni bancarie. Solitamente questo tipo di transazioni corrotte avviene in numero molto limitato, si supponga di avere una popolazione di record in cui il 99% appartiene alla classe di maggioranza (transazioni sicure). Se si utilizza un modello che classifica ogni nuova transazione come "sicura" allora si otterrà un accuracy del 99% (oppure un error rate del 1%). Chiaramente non si può definire il modello come buono poiché ogni transazione corrotta non verrà identificata portando a molti problemi finanziari.

È prassi raccogliere le informazioni della classificazione di un modello in una tabella chiamata *Confusion Matrix*. Sotto è rappresentata una Confusion Matrix di un problema di classificazione binaria, è però possibile estendere questo concetto anche a problemi *multi-class*.

- True Positive (TP): numero di record positivi classificati correttamente come positivi.
- False Positive (FP): numero di record negativi classificati erroneamente come positivi.
- True Negative (TN): numero di record negativi classificati correttamente come negativi.
- False Negative (FN): numero di record positivi classificati erroneamente come negativi.

		Classe Predetta	
		-	+
Classe Reale	-	*TrueNegative*	*FalseNegative*
	+	*FalsePositive*	*TruePositive*

A partire dai valori della matrice è possibile estrarre informazioni e metriche più significative rispetto al modello sviluppato:

Sensibilità: Nota anche come *TPR (True Positive Rate)* o *Recall*. Indica la frazione dei record positivi classificati correttamente positivi, viene molto utilizzata in ambito medico. Considerando un classificatore come un test diagnostico, la sensibilità rappresenta la proporzione di persone malate che effettivamente vengono etichettate come tali. Può essere vista come la capacità del classificatore di identificare record positivi. Se il classificatore ha elevata sensibilità allora un test negativo suggerisce l'assenza di malattia. Ad esempio se un classificatore ha sensibilità pari al 100% allora riconosce tutte le persone malate, di conseguenza se una persona viene etichettata negativamente allora sarà quasi sicuramente sana visto che se fosse stata malata il test l'avrebbe riconosciuta.

$$TPR = \frac{TP}{TP + FN}$$

Specificità: Nota anche come *TNR (True Negative Rate)*. Indica la frazione dei record negativi classificati correttamente come negativi. Considerando un problema medico rappresenta la proporzione di persone sane che il classificatore identifica come tali. Esprime la capacità del classificatore di identificare risultati negativi. Se un test ha un'elevata specificità allora un risultato positivo dal test indica elevata probabilità che la persona sia malata.

$$TNR = \frac{TN}{TN + FP}$$

FalsePositiveRate: Indica la frazione dei record negativi classificati come positivi

$$FPR = \frac{FP}{FP + TN}$$

FalseNegativeRate: Indica la frazione dei record positivi classificati come negativi

$$FNR = \frac{FN}{FN + TP}$$

Precision: Essa indica la frazione dei *True Positve* rispetto a tutti i risultati positivi. Maggiore è la precision meno sono i falsi positivi commessi dal modello, un elevato *recall* indica invece che i falsi negativi sono limitati. Tra *sensibilità* e *precision* vi è una sottile differenza: la prima ad esempio indica la percentuale di pazienti identificati come malati tra tutti quelli che lo sono nella realtà, mentre la seconda indica la percentuale di pazienti realmente malati tra tutti quelli etichettati come malati dal classificatore.

$$Precision = \frac{TP}{TP + FP}$$

2.6.1 Valutazione dell'accuratezza di un classificatore

Esistono diverse tecniche per valutare l'accuratezza che un algoritmo di costruzione di un modello può garantire. Solitamente le performance di un modello vengono valutate su un insieme di dati diverso (test set) da quello utilizzato per la costruzione, in esso sono comunque note le classi di appartenenza dei vari record. In questo modo si ottiene una misura delle prestazioni priva di bias in quanto eseguita su istanze non utilizzate in fase di apprendimento. Spesso queste tecniche vengono utilizzate per determinare, a parità di algoritmo di costruzione, quali parametri forniscono il modello migliore. È importante sottolineare che il modello finale viene sempre costruito utilizzando l'intero dataset, per cui la stima di accuratezza che si ottiene risulta conservativa.

Esistiono diversi metodi [6] che permettono di stimare l'accuratezza di un algoritmo di induzione: *Hold Out* in cui il dataset viene suddiviso in training set e test set, solitamente

con cardinalità rispettivamente 2/3 ed 1/3, oppure *Random Subsampling* che equivale alla media dei risultati ottenuti con l'applicazione dell' Hold Out k volte con training set e test set diversi. Il metodo sicuramente più utilizzato è la *Cross Validation*: a partire dall' insieme di record originale E si creano k sottoinsiemi tali che:

$$E = \bigcup E_i \text{ dove } |E_i| = {}^{|E|}/_{k};\ E_i \cap E_j = \varnothing\ \forall(i,\, j) \text{ con } i \neq j$$

Viene costruito un modello M_i per ogni training set $E \setminus E_i$ e viene validato sull'insieme E_i. Il procedimento viene ripetuto k volte in modo tale che ciascun E_i sia utilizzato esattamente una volta come test set. Le performance finali sono date come la media tra tutti i modelli creati.

$$M_i = \text{modello costruito da } E \setminus E_i$$

$$accuracy = \frac{\sum_{i=1}^{k} |\{t \in E_i\ :\ M_i(t) = t(C)\}|}{|E|}$$

I vari insiemi E_i prendono il nome di *fold* e si parla di *k-fold Cross Validation*. Nel caso in cui venga mantenuta la proporzione tra il numero dei record delle diverse classi all'interno dei singoli fold si parla di *Stratified Cross Validation*.

La Cross Validation prevede che le varie fold siano create scegliendo a caso i record a disposizione. In ambito medico spesso i dati che provengano da cliniche o centri diversi, questo accade anche nel dataset usato in questo lavoro (Padova, Reggio Emilia, Bologna, etc.), perciò utilizzando la Cross Validation si hanno delle fold composte da pazienti riferiti a centri differenti. Questa situazione introduce del bias nella validazione del modello e porta a sovrastimare le prestazioni. Per ottenere dei risultati più veritieri si è creata una nuova validazione che è stata chiamata *Leave One Out Clinica* ($LOOC$). Si tratta di una generalizzazione dell Cross Validation dove all'interno di ciascuna fold vengono inserite solamente le istanze appartenenti alla medesima clinica. Il procedimento di validazione resta uguale: la fold i contiene tutti i record del centro i che verranno usati una volta per la validazione ed *n-1* volte per la costruzione del modello (n rappresenta il numero di cliniche presenti nel dataset). Operando in questo modo si valuta il modello cercando di rispettare il più possibile le caratteristiche reali dei dati. Se per esempio ci sono delle differenze, anche marginali, tra le caratteristiche dei pazienti di Padova e quelli di Reggio Calabria (e.g nel nord del paese la popolazione ha una pressione sanguigna mediamente più bassa/alta rispetto al sud) è giusto valutare il modello utilizzando solo record provenienti dallo stesso centro, altrimenti si compromette la fase di validazione rendendola poco realistica. Con la Cross Validation, in situazioni del genere, scegliendo i recordi in modo casuale si mettono assieme pazienti con caratteristiche intrinseche diverse (dovute al fatto di provenire da centri diversi). Un confronto tra la validazione 10-Fold Cross Validation e Leave One Out Clinica può essere osservato nell'Appendice B.

2.6.2 Curve ROC

L'uso dei PET permette di utilizzare altre metriche di prestazione rispetto ai tradizonali decision tree. Si consideri un problema binario, definendo una soglia di probabilità per una classe è possibile etichettare in maniera standard i record e ricorrere alle metriche di valutazione presentate in sopra. Quando l'output di un modello fornisce un valore di probabilità è possibile ricorrere all'uso delle curve *ROC* per la valutazione della bontà di quest'ultimo. ROC sta per *Receiver Operating Characteristics* ed è una tecnica per visualizzare, organizzare e selezionare classificatori sulla base delle loro performance. In ambito medico è utilizzata da molto tempo ma recentemente è stata introdotta anche nel campo del Machine Learning [14]. Solitamente queste curve vengono utilizzate nei problemi di classificazione binaria ma i loro concetti possono essere estesi anche a problemi multi-class come spiegato in [15]. Nel seguito si considererà un problema con due classi.

Una curva ROC è un grafico bidimensionale dove il True Positive Rate è plottato lungo asse Y e il False Positive Rate lungo X. Si ricorda che:

$$TPR = \frac{TP}{TP + FN} \text{ e } FPR = \frac{FP}{FP + TN}$$

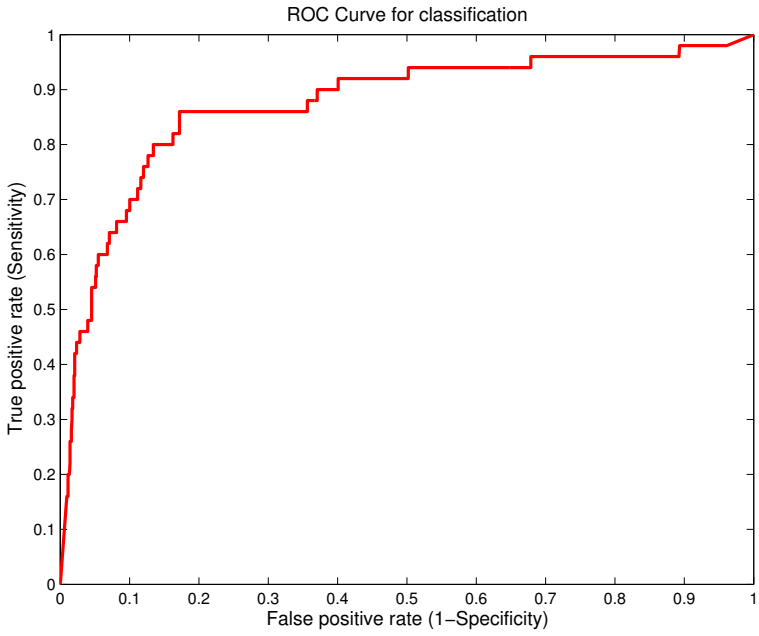


Figura 2.8: Esempio di curva ROC

Come detto precedentemente la sensibilità coincide con il TPR e la specificità è 1-FPR, allora la ROC lega la probabilità di ottenere un risultato vero-positivo nella classe dei malati-veri (ossia la sensibilità) alla probabilità di ottenere un risultato falso positivo nella classe dei non-malati (ossia 1-specificità). In altre parole, vengono studiati i rapporti fra allarmi veri (hit rate) e falsi allarmi.

Un classificatore discreto produce in output solamente le class label perciò si ha solamente una coppia (FPR,TPR) che coincide in un solo punto nello spazio ROC.

Ci sono diversi punti importanti nello spazio ROC:

(0,0) Rappresenta una strategia per la quale non vi è mai una classificazione positiva; in questo modo non ci sono errori di Falsi Positivi, ma tantomeno neanche Veri Positivi.

(1,1) Rappresenta la strategia opposta in cui vi è sempre una classificazione positiva, questo porta a non avere Falsi Negativi, ma chiaramente neanche ad avere Veri Negativi dato che nessuna istanza è classificata negativamente.

(0,1) Rappresenta la classificazione perfetta.

La linea diagonale $y = x$ rappresenta una strategia di classificazione random. Ad esempio un classificatore che metà delle volte etichetta in maniera positiva e metà in modo negativo avrà un punto $(0.5\,,\,0.5)$ nella ROC. Ogni classificatore che appare nel triangolo basso (al di sotto della retta $y = x$) ha delle performance peggiori di un classificatore random, solitamente questa zona è vuota.

Fino ad ora si è parlato di singoli punti all'interno dello spazio ROC, in realtà avendo a disposizione delle probabilità si possono costruire delle curve vere e proprie come quella mostrata in Figura 2.8. L'idea è quella di ordinare le probabilità delle istanze in maniera crescente, dopodiché si scelgono delle soglie incrementali (e.g 5%, 10%, 15%, ...) e si vanno a valutare TPR e FPR considerando come positivi tutti i record la cui probabilità di appartenere alla classe positiva è superiore alla soglia scelta. Se ad esempio la soglia è 0, allora tutte le istanze sono considerate positive e si avrà un punto in (1, 1) nella ROC poiché tutti i positivi sono identificati (TPR=1), ma allo stesso tempo tutti i negativi sono considerati positivi e FPR=1. Alzando la soglia, se esiste un ordinamento coerente con le classi, ci si aspetta di abbassare il FPR mentre il TPR rimarrà ad 1 fino a che non vi è un vero positivo al di sotto della soglia scelta. Così facendo si ottiene una curva spezzata, cioè una serie di step, è possibile eliminare la scalettatura utilizzando un'interpolazione.

La curva ROC può essere analizzata in maniera qualitativa andando ad osservare quanto essa sia schiacciata rispetto all'asse Y oppure rispetto alla retta: TPR=1. Nel primo caso significa che il modello riesce ad identificare le istanze negative associandovi basse probabilità, mentre nel secondo caso riescono a venire identificate le istanze positive e ad

esse vengono associate elevate probabilità. In termini generali una curva è tanto migliore quanto riesce a rimanere schiacciata rispetto al punto (0,1).

Per comparare due classificatori può essere utile ridurre la ROC ad un singolo valore scalare che rappresenta le performance. Un metodo comune è quello di calcolare l'AUC: *Area Under the roc Curve*. Dato che si tratta di una porzione di un'area di un quadrato di lato unitario essa ha sempre un valore compreso tra 0 ed 1. L'AUC ha un'importante proprietà statistica: è equivalente alla probabilità che il classificatore assegni un punteggio maggiore ad un'istanza positiva scelta a caso rispetto ad una negativa scelta a caso. Questo è equivalente al Wilcoxon Test of Ranks [15].

Le curve ROC vengono utilizzate anche in fase di ottimizzazione, ovvero è possibile scegliere il valore di cut-off per la probabilità che garantisce i valori di specificità e sensibilità desiderati e richiesti dal problema.

Un'utile proprietà delle curve ROC è quella di essere insensibili alla variazione della grandezza dell'insieme dei record utilizzati. Se la proporzione tra record positivi e record negativi cambia in un test set allora la curva non varia. Questo è dovuto al fatto che la ROC è basata su TPR e FPR, ovvero su dei valori che dipendono dal rapporto tra gli elementi di una colonna della Confusion Matrix e non dalla distribuzione delle classi. In ambiti in cui vi può essere un forte cambiamento della distribuzione delle classi (e.g l'incidenza di una malattia può aumentare drasticamente nel corso degli anni) questa si rivela una proprietà molto utile. Per tutti i motivi elencati le curve ROC sono uno strumento molto potente ed utile anche nel campo del Machine Learning.

ROC & Cross Validation

Può sorgere spontaneo chiedersi come si possano utilizzare le curve ROC quando si ricorre alla Cross Validation per determinare le prestazioni di un classificatore. Come citato in 2.6.1 non conviene valutare il modello sullo stesso insieme di training usato per l'induzione, lo stesso vale nell'uso delle curve ROC. Brevemente si ricorda che nella Cross Validation si suddivide il dataset E in k sottoinsiemi di taglia $|E|/k$, iterativamente si utilizza l'insieme $E \setminus E_i$ per la costruzione del modello e l'insieme E_i per la validazione. Utilizzando i PET ad ogni istanza degli E_i è associato un valore di probabilità relativo alle varie classi. In [15] vengono illustrate alcune tecniche su come utilizzare gli scores dei vari sottoinsiemi E_i per costruire una curva ROC. Una prima idea molto semplice è quella di mettere assieme tutti i recor degli E_i in un nuovo insieme E' sul quale si può applicare l'algoritmo spiegato sopra. Il primo obiettivo di creare test set multipli con la Cross Validation è quello di studiare la varianza dei risultati, però questa semplice operazione di fusione non permette di farlo. Un metodo alternativo è quello di creare un curva ROC per ciascun sottoinsieme E_i, queste curve vengono poi utilizzate in un processo di mediazione che porta alla costruzione di

una ROC finale. Ad esempio il *Vertical Averaging* prende campioni dalle curve ROC per valori fissi di FPR e media i relativi valori di TPR.

Le stesse tecniche possono venire utilizzate qualora si vogliano eseguire più Cross Validation sullo stesso dataset in modo da evitare di ottenere dei risultati che dipendono da come i record vengono distribuiti all'interno delle varie fold in maniera random.

2.7 Trattamento dei Missing Values

Quando si ha a che fare con dati provenienti dal mondo reale è impossibile non imbattersi nel problema dei dati mancanti. Molto spesso i record presentano alcune variabili per le quali non è noto il valore: le persone non sono in grado di rispondere a tutte le domande di un questionario, i sensori non sono buoni, ci sono malfunzionamenti degli strumenti di misurazione, alcuni test medici non possono venire eseguiti per ragioni diverse, etc. Le performance di un sistema sono strettamente legate alla percentuale di missing values: un valore inferiore al 5% viene considerato solitamente gestibile, valori dal 5% al 15% richiedono metodi ad hoc, mentre valori superiori al 15% sono difficilmente gestibili [16]. Un'idea può essere quella di sostituire i valori mancanti con:

- La media dei valori della feature su tutto il training set
- Il valore presente nei record più simili
- Il valore rispetto ai *k-Nearest Neighbours*

In letteratura esistono molte tecniche che permettono di gestire il problema dei missing values [17], l'algoritmo C4.5 prevede invece un particolare metodo di gestione. Quando viene effettuato lo split su un determinato attributo A esso viene realizzato considerando solamente i record di E che hanno un valore noto per A. Quando lo split viene realizzato ($E = \bigcup_i E_i$) il suo valore informativo viene ridotto di un fattore m pari alla frazione dei record che hanno un missing value per l'attributo A. Lo *SplitInfo* viene poi calcolato considerando *k+1* sottoinsiemi, ovvero si considera un sottoinsieme anche per i record con valore mancante

$$\triangle_{info} = m \cdot \left(H(E) - \sum_{j-1}^{k} \frac{|E_j|}{|E|} \cdot H(E_j) \right)$$

Una volta che è stato definito lo split, ciascun record viene associato ad un sottoinsieme E_i con un punteggio diverso che è proporzionale alla probabilità di appartenenza ad esso. Ipotizzando che lo split abbia k possibili uscite allora il punteggio di un istanza t per la quale $t[A]$ non è noto rispetto al sottoinsieme E_i è:

$$w_i = \hat{P}(t\,[A] = v_i) = \frac{|\{t \in E : t\,[A] = v_i\}|}{|E|}$$

dove v_i con $1 \leq i \leq k$ sono le uscite del test.

Utilizzando questo procedimento può capitare che un record t, in una fase di costruzione dell'albero, abbia già un punteggio assegnato visto che E può essere la partizione di un insieme più grande. Quando per il record t deve essere scelta una partizione E_i dovuta ad uno split rispetto ad un nuovo attributo B, per il quale non è noto il valore, allora si aggiorna il punteggio:

$$w_i \leftarrow w_i \cdot \hat{P}(t\,[B] = v_i) = w_i \cdot \frac{|\{t \in E : t\,[B] = v_i\}|}{|E|}$$

In questo modo è possibile utilizzare nella costruzione dell'albero tutti i record appartenenti al training set originale, la somma delle cardinalità delle foglie coincide con quella di un albero costruito rispetto ad un training set senza valori mancanti.

Un approccio simile si può utilizzare per classificare un record con valori mancanti rispetto a determinati attributi. In presenza di un test per il quale il record non ha valore vengono esplorate tutte le possibili uscite, visto che a causa di questo si possono avere cammini multipli, i possibili risultati vengono combinati in maniera aritmetica. Per il record t viene calcolata la sua distribuzione di probabilità rispetto a tutte le classi C considerando tutte le foglie in cui esso potrebbe finire seguendo scelte diverse nei vari test. L'istanza viene infine etichettata considerando quale classe risulta più probabile.

2.7.1 EM: Expectation-Maximization

A differenza dei Decision Tree, ci sono tecniche di apprendimento automatico (e.g SVM) che non prevedono la possibilità di gestire istanze con valori mancanti, perciò il training set T utilizzato nella fase di learning deve essere completo. Visto che i dataset provenienti dal mondo reale sono spesso ricchi di missing values è necessario applicare un metodo per completare il dataset. Una soluzione prende il nome di *EM Algorithm* (Expectation-Maximization), è stata presentata per la prima volta nel 1977 [18]. Si tratta di un algoritmo iterativo per la stima di parametri che utilizza il criterio della *massima verosimiglianza* (*maximum likelihood*) quando alcune delle variabili coinvolte non sono osservabili (e.g missing o incomplete values). L'algoritmo formalizza un'idea intuitiva per gestire i missing values:

1. Stima i parametri iniziali utilizzando i dati completi
2. Utilizza i parametri per "stimare" i dati mancanti

3. Utilizza i "nuovi" dati per aggiornare i parametri

4. Ripete da 2) fino alla convergenza

In parole povere i dati vengono visti come una matrice incompleta, di essa si cerca di determinare alcuni parametri quali media e matrice di covarianza. Con tali parametri si cerca di stimare i valori per sostituire i missing values. Completata questa fase si calcolano nuovamente i parametri utilizzando questa volta tutta la matrice e, nel caso ci sia una significativa differenza rispetto al passo precedente, si aggiornano i valori dei missing values. Viene ora data una breve spiegazione matematica dell'algoritmo.

Sia $X \in \Re^{n \times p}$ la matrice dei dati con n record ciascuno dei quali con p variabili, vi siano alcuni valori mancanti nelle righe. L'idea è di stimare la media $\mu \in \Re^{1 \times p}$ e la matrice di covarianza $\Sigma \in \Re^{p \times p}$ delle variabili a partire dalla matrice incompleta. Per un record $\mathbf{x_i}$ con missing values sia il vettore $\mathbf{x_a}$ formato dalle p_a variabili per le quali i valori sono disponibili, sia invece $\mathbf{x_m}$ un vettore formato dalle p_m rimanenti variabili per le quali non sono noti i valori. Si suddivida la media μ in due parti: μ_a contenente le medie delle variabili per le quali il vettore $\mathbf{x_i}$ ha un valore disponibile, mentre μ_m sia formato dalle medie delle variabili per le quali $\mathbf{x_i}$ ha un missing value. Per ogni record $\mathbf{x_i}$ con dei missing values la relazione tra le variabili con valori mancanti e le variabili con valori disponibili è modellata da un modello di regressione lineare:

$$\mathbf{x_m} = \mu_m + (\mathbf{x_a} - \mu_a)\,\mathbf{B} + \mathbf{e} \tag{2.7}$$

La matrice $\mathbf{B} \in \Re^{p_a \times p_m}$ è la matrice dei coefficienti di regressione, mentre il residuo $\mathbf{e} \in \Re^{1 \times p_m}$ è un vettore casuale a media nulla e matrice di covarianza non nota $\mathbf{C} \in \Re^{p_m \times p_m}$. In ciascuna iterazione EM usa le stime della media μ e della matrice di covarianza Σ per stimare le matrici $\mathbf{B}$ e $\mathbf{C}$ per ogni record con missing values. Usando il modello di regressione per ciascun record, i missing values sono riempiti con i valori calcolati da 2.7 , è quindi possibile poi effettuare una nuova stima di μ e Σ a partire dal dataset completo.

Siano $\hat{\mu}^{(t)}$ e $\hat{\Sigma}^{(t)}$ le stime della media e della matrice di covarianza nell' t-sima iterazione dell'algoritmo. Queste stime sono il risultato della precedente iterazione, nel caso del passo iniziale esse possono venire calcolate riempiendo i valori mancanti con valori iniziali "intuitivi". Per un record $\mathbf{x_i}$ con valori mancanti sia $\hat{\Sigma}^{(t)}$ la stima della matrice di covarianza partizionata secondo la divisione delle variabili con valore disponibile e non dello stesso record: perciò si ha $\hat{\Sigma}^{(t)}_{aa}$ la sottomatrice della matrice di covarianza stimata $\hat{\Sigma}^{(t)}$ che riguarda varianza e covarianza delle variabili che hanno valori disponibili. Allo stesso modo si definisce la sottomatrice $\hat{\Sigma}^{(t)}_{mm}$ per le variabili con missing values. Siano poi $\hat{\Sigma}^{(t)}_{am}$ e $\hat{\Sigma}^{(t)}_{ma}$ con $\hat{\Sigma}^{(t)}_{am} = \hat{\Sigma}^{(t)}_{ma}$ le matrici di cross-covarianza tra le variabili a valori disponibili con le variabili prive di valori. Nota $\hat{\Sigma}^{(t)}$ allora la massima verosimiglianza condizionata stimata dei coefficienti di regressione vale: $\hat{\mathbf{B}} = \hat{\Sigma}^{-1}_{aa}\hat{\Sigma}_{am}$. Dalla struttura del modello di

regressione 2.7 segue che, a partire da $\hat{\mathbf{B}}$ e della matrice di covarianza partizionata $\hat{\Sigma}^{(t)}$ allora la matrice di covarianza residua prende la forma generica di:

$$\hat{\mathbf{C}} = \hat{\Sigma}_{mm} + \hat{\mathbf{B}}^{\mathrm{T}}\hat{\Sigma}_{aa}\hat{\mathbf{B}} - \hat{\mathbf{B}}^{\mathrm{T}}\hat{\Sigma}_{am} - \hat{\Sigma}_{ma}\hat{\mathbf{B}}$$

sostituendo $\hat{\mathbf{B}}$ si arriva a $\hat{\mathbf{C}} = \hat{\Sigma}_{mm} - \hat{\Sigma}_{ma}\hat{\Sigma}_{aa}^{-1}\hat{\Sigma}_{am}$.

Il valore atteso condizionato $\hat{\mathbf{x}}_m = E\left(\mathbf{x}_m \mid x_a;\, \hat{\mu}^{(t)},\, \hat{\Sigma}^{(t)}\right)$ dei missing values in un particolare record deriva da $\hat{\mathbf{B}}$ e dai valori disponibili $\mathbf{x_a}$:

$$\hat{\mathbf{x}}_\mathbf{m} = \hat{\mu}_m + (\mathbf{x_a} - \hat{\mu}_a)\,\hat{\mathbf{B}}$$

dove il vettore $\hat{\mu}_a$ è la parte della stima della media $\hat{\mu}^{(t)}$ riferita alle variabili con valori noti mentre $\hat{\mu}_m$ si riferisce alle variabili con missing values.

Una volta che i valori mancanti dei record $\mathbf{x_i}$ sono stati riempiti con i valori di $\hat{x}_m$ allora si può ricalcolare la stima della media del "nuovo" dataset completo: $\hat{\mu}^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x_i}$. Allo stesso modo è possibile trovare nuovamente la stima della matrice di covarianza, essa deriva dalla aspettazione condizionata dei prodotti incrociati $\hat{\mathbf{S}}_i^{(t)} \equiv E\left[\mathbf{x_i^T x_i} \mid \mathbf{x}_a;\, \hat{\mu}^{(t)},\, \hat{\Sigma}^{(t)}\right]$ cioè:

$$\hat{\Sigma}^{(t+1)} = \frac{1}{\widetilde{n}}\sum_{i=1}^{n}\left\{\hat{\mathbf{S}}_i^{\,(t)} - \left[\hat{\mu}^{(t+1)}\right]^{\mathrm{T}}\hat{\mu}^{(t+1)}\right\}$$

il fattore di normalizzazione $\widetilde{n}$ è dato dal numero di gradi di libertà della matrice di covarianza del dataset.

L'iterazione successiva di EM inizia con le nuove stime $\hat{\mu}^{(t+1)}$ e $\hat{\Sigma}^{(t+1)}$. L'algoritmo si ferma quando converge, ovvero quando $\hat{\mu}^{(t)}$ e $\hat{\Sigma}^{(t)}$ o i valori calcolati $\hat{\mathbf{x}}_\mathbf{m}$ smettono di cambiare significativamente. Per lo sviluppo di questa tesi è stata utilizzata una versione leggermente diversa dell'algoritmo chiamata *EM regularized* [19], gli step di esecuzione sono identici solamente che $\hat{\Sigma}_{aa}^{(t)}$ viene rimpiazzata da un altra matrice. Non vengono qui riportati ulteriori dettagli.

2.8 Trattamento di Dataset Sbilanciati

Sono molti i dataset che rappresentano ambiti del mondo reale in cui è presente uno sbilanciamento del numero di record appartenenti alle diverse classi. Esempi di questa situazione sono: l'identificazione di frode in conti bancari, la text classification, la diagnosi medica, l'intrusione in reti informatiche, etc. In ciascuno di questi ambiti una classe è presente in numero molto maggiore rispetto ad un'altra. Tecnicamente parlando ogni dataset che

presenta una distribuzione non equa tra le classi può essere definito "*imbalanced*" o sbilanciato. Non è impossibile trovarsi di fronte a sbilanciamenti del tipo 100:1, 1000:1 e 10000:1. Ci possono essere degli ambiti in cui lo sbilanciamento dei dati è *intrinseco* al problema stesso, ad esempio il numero delle persone sane rispetto a quelle affette da una malattia rara. In altri casi si può avere a che fare con uno sbilanciamento *estrinseco* non dovuto alla natura dei dati ma, ad esempio, causato dalla raccolta non uniforme di essi (e.g. può essere molto costoso reperire informazioni riguardo ad una particolare classe). C'è poi da considerare il concetto di *imbalance relativa*: si supponga di avere un dataset con 100000 record e che ci sia un rapporto tra le classi di 100:1, ci si aspetta quindi di avere solamente 1000 record della classe di minoranza. Se si raddoppia lo spazio di ricerca dei dati si otterrà la stessa distribuzione tra le classi ma quella di minoranza avrà ora 2000 record, chiaramente la classe minoritaria ha ancora cardinalità più bassa ma 2000 possono essere sufficienti ad un classificatore per ottenere buoni risultati. La classe di minoranza non è di per se rara ma lo è se la si considera rispetto a quella di maggioranza.

Negli anni l'interesse della comunità scientifica riguardo a questo problema è cresciuto, vi sono state numerose conferenze di discussione [20] [21]; ciò è dovuto anche al fatto che l'imbalance dei dati può provocare performance sub-ottimali, sopratutto in presenza di un numero di istanze limitato per una particolare classe.

Diverse sono le soluzioni che sono state proposte nel tempo: esse possono essere applicate a livello di algoritmo di costruzione del modello oppure direttamente sui dati. In seguito vengono presentate le tecniche che sono state utilizzate in questo lavoro per cercare di arginare il fenomeno dell'imbalance, una di esse va a lavorare a livello di algoritmo (Cost-Sensitive Learning) mentre le altre due effettuano un lavoro di preprocessing sui dati (Resampling e Feature Selection).

2.8.1 Approccio Cost-Sensitive

In molte situazioni gli errori dovuti alla classificazione errata non possono essere considerati in maniera uguale. Ci sono casi in cui il costo legato agli errori è costante, sbagliare nel classificare una classe oppure un'altra non varia la situazione generale. Si prenda però l'ambito medico, ad esempio un test per l'identificazione del cancro (classe positiva). È chiaro che se il test non individua il cancro in un individuo malato (Falso Negativo) è molto più grave e costoso rispetto ad un Falso Positivo dove ad una persona sana viene diagnosticata la malattia. Il paziente può avere delle complicazioni a causa di un errore, allo stesso modo è più rischioso "mancare" un terrorista che porta una bomba con se piuttosto che perquisire un innocente all'interno di un aereoporto.

Si possono quindi associare dei costi a ciascun tipo di errore utilizzando una *Matrice dei Costi* che ha la stessa struttura della Matrice di Confusione. Nella matrice in Figura 2.9 si vede che un Falso Positivo costa 5 mentre un Falso Negativo costa 6 volte più rispetto

all'errore nella classificazione di un record negativo. Come si tiene conto però di questi costi in fase di classificazione?

		Classe Predetta	
		-	+
Classe Reale	-	1	30
	+	5	1

Figura 2.9: Esempio di Matrice dei Costi per un problema lineare

Ci sono due modi per utilizzare i costi di classificazione: *Cost-sensitive Classification* e *Cost-sensitive Learning*. Nel primo caso i costi sono ignorati in fase di costruzione del modello e vengono presi in considerazione solamente in fase di valutazione. Se ad esempio il modello fornisce in output la probabilità associata a ciascuna istanza, essa può essere aggiustata in modo tale da minimizzare i costi attesi della predizione. Dato un insieme di probabilità per una certa istanza di test solitamente viene scelto il valore più verosimile, con questo metodo si va a predire cercando di minimizzare i costi dovuti agli eventuali errori di classificazione. Si supponga di avere a che fare con un problema nel quale sono presenti tre classi: *a, b, c* dove p_a, p_b, p_c indicano le probabilità stimate relative ad esse di un'istanza di test; la matrice dei costi è definita sotto in Figura 2.10. Se il modello predice a allora il costo atteso di questa predizione è ottenuto moltiplicando la prima colonna della matrice [0 1 1] con il vettore $[p_a\, p_b, p_c]$, questo porta al valore $p_b + p_c$. Allo stesso modo i costi per predire le altre due classi sono $1 - p_b$ e $1 - p_c$. Per questa matrice dei costi scegliere la predizione con il minor costo coincide con lo scegliere quella con maggiore probabilità, usando una matrice diversa questo non è più vero.

		Classe Predetta		
		a	b	c
Classe Reale	a	0	1	1
	b	1	0	1
	c	1	1	0

Figura 2.10: Esempio di Matrice dei Costi per un problema ternario

Nel Cost-Sensitive Learning i valori espressi dalla matrice dei costi vengono presi in considerazione durante la fase di costruzione del modello. Solitamente viene fatto un ri-bilanciamento delle istanze del training set assegnando un peso basato sui costi delle loro classi di appartenenza. In fase di apprendimento del modello, visto che si vogliono minimizzare i costi degli errori, si faranno scelte diverse a seconda se si utilizza un peso unitario per tutto il training set oppure se si usano i pesi precedentemente definiti. Se in un problema binario le istanze della classe positiva hanno un peso cinque volte maggiore rispetto alla classe negativa, visto che le scelte impongono di minimizzare il peso degli errori, si otterrà un modello "sbilanciato" che cercherà di evitare quelli sulle istanze positive piuttosto che su quelle negative.

Cost-Sensitive Tree Induction

Vengono ora discussi i due approcci presentati sopra applicati ad un albero decisionale.

Sia N il numero totale delle istanze di un training set T ed N_j il numero delle istanze di classe j. Siano $N(v)$ e $N_j(v)$ rispettivamente il numero delle istanze nel nodo v ed il numero delle istanze di classe j nel medesimo nodo. Si definisca $C(j)$ come il costo di classificazione errata della classe$j$. Prendendo un caso generale una possibile forma di $C(j) = \sum_i cost(i,\, j)$ definita in [5] dove: $cost(i,\, j)$ indica il costo di assegnare un'istanza della classe j come appartenente alla classe i. In caso di costi unitari si assuma che il peso di ciascuna istanza sia unitario. L'idea è quella di modificare il peso di ciascuna istanza in modo proporzionale ai costi di classificazione errata della relativa classe mantenendo però la somma dei pesi sempre uguale ad N. Per cui il peso di un record può essere definito:

$$w(j) = C(j) \cdot \frac{N}{\sum_i C(i) \cdot N_i}$$

così la somma di tutti i pesi vale: $\sum_j w(j) N_j = N$.

La probabilità che un'istanza che viene mappata nel nodo v appartenga alla classe j è:

$$p(j \mid v) = \frac{w(j) \cdot N_j(v)}{\sum_i w(i) \cdot N_i(v)}$$

Gli algoritmi di induzione degli alberi (e.g C4.5) possono essere utilizzati senza alcuna modifica. La nuova stima di probabilità definita va sostituire il rapporto $N_j / \sum_i N_i(v)$ nella valutazione del migliore split. Ad esempio utilizzando l'Entropia come misura: $-\sum_j p(j \mid v) \cdot \log\left[p(j \mid v)\right]$ il valore $p(j \mid v)$ viene calcolato secondo la nuova formula proposta. Si tratta perciò di un approccio Cost-Sensitive Learning in cui i costi incidono nella costruzione dell'albero.

In alternativa si può definire un "peso di classe" come la somma dei singoli pesi delle istanze appartenenti alla medesima classe: $W_j(v) = \sum_j w(j)$. Si può utilizzare questo valore in sostituzione della frequenza $N_j(v)$ per calcolare le probabilità:

$$p(j \mid v) = \frac{W_j(v)}{\sum_i W_i(v)}$$

Il vantaggio di questo secondo approccio è che l'intero processo di costruzione dell'albero è lo stesso che avviene utilizzando gli algoritmi tradizionali: siamo quindi in presenza di una Cost-Sensitive Classification.

2.8.2 Features Selection

Quando si ha a che fare con dataset che riguardano molti campi del mondo reale (bioinformatica, image processing, text classification) bisogna porre l'attenzione sulla dimensionalità dei dati, ovvero sul numero delle feature disponibili. Nell'ambito del machine learning vi è un concetto noto come *Curse of Dimensionality* che afferma:

In assenza di ipotesi semplificative la dimensione del training set necessario per stimare una funzione di alcune variabili ad un certo livello di accuratezza cresce esponenzialmente con il numero di variabili.

Cercare di sviluppare un modello a partire da un dataset ad alta dimensionalità può essere molto costoso in termini computazionali e può portare a risultati non accurati. Spesso invece risulta interessante identificare un sottoinsieme di feature che siano più predittive rispetto all'insieme di partenza, inoltre alcune variabili inutili possono comportarsi in maniera simile al rumore ed impattare in modo negativo nella classificazione. L'uso della feature selection è spesso utile in molti problemi di Data Mining, la sua importanza assume maggiore rilevanza in una situazione di sbilanciamento dove scegliere le variabili che più separano le classi porta ad una migliore prestazione [22].

Per risolvere questi problemi esistono due tecniche: *Feature Trasformation* e *Feature Selection*. La prima cerca di trasformare le feature originali in un nuovo spazio mantenendo più informazione possibile. Ci sono due tipologie:

- Feature Extraction: Prevede un processo di mapping dei dati dallo spazio originale ad uno nuovo sottodimensionato $x \in R^d \rightarrow x' \in R^p \,\mathrm{con\,p} \ll \mathrm{d}$. La tecnica più usata che porta anche ai risultati migliori è nota come PCA (Principal Component Analysis) [23].

- Feature Generation: cerca di scoprire la mancanza d'informazioni che può esserci nei dati a partire dalle feature del dataset originale e prova a creare un nuovo spazio a dimensione maggiore in cui sono presenti variabili artificiali che cercano di colmare la lacuna scoperta.

La Feature Generation crea delle nuove variabili con l'obiettivo di fornire informazioni utili alla classificazione, ovviamente in questo modo si aumenta la dimensionalità del dataset rischiando di aumentare il curse of Dimensionality citato prima. Per questo è conveniente far seguire alla Feature Generation delle tecniche di Feature Selection che permettano di individuare il sottoinsieme di variabili più significative.

Gli algoritmi di *Features Selection* (FS) prevedono un approccio diverso cercando di localizzare un sottoinsieme minimo ottimale di variabili originali piuttosto che trasformarle in un nuovo spazio. Per lo scopo del Knowledge Discovery interpretare l'output di algoritmi di Features Extraction può essere problematico dato che le nuove variabili in alcuni casi non hanno un significato "fisico" nel dominio di lavoro, con la FS questa difficoltà non si pone.

Le tecniche di selezione tipicamente prevedono: una strategia di ricerca per esplorare lo spazio dei sottoinsiemi di feature (compresi metodi per determinare un punto di partenza adatto e generare sottoinsiemi candidati successivi) ed un criterio di valutazione per valutare e analizzare i candidati. Queste tecniche possono essere divise in due grandi categorie: Filtri e Wrapper.

Filtri

I filtri tentano di rimuovere gli attributi irrilevanti dall'insieme delle feature prima dell'esecuzione dell'algoritmo di learning. I dati sono analizzati per identificare le dimensioni che sono più rilevanti per discriminare la struttura. Il sottoinsieme scelto è poi utilizzato per allenare il modello. Centrale in questa tecnica è il criterio usato per assegnare il punteggio rispetto alla predittività dell'attributo. Uno dei metodi più noti ed utilizzati è l'*Information Gain* (IG) presentata nella Sezione 2.2.4, si tratta di una misura della quantità di informazione che la feature porta al training set [3]. È definita come la riduzione attesa dell'entropia causata dalla partizione del training set T utilizzando la feature f. I vari sottoinsiemi T_v sono costruiti andando a suddividere T sulla base dei valori v del dominio di f, in presenza di attributi numerici è quindi necessaria una discretizzazione

$$IG(T,\, f) = H(T) - \sum_{v \in values(f)} \frac{|T_v|}{|T|} \cdot H(T_v)$$

Si ricorda che l'entropia è una misura di quanta impurità c'è nel dataset, magiore è meno informazioni si hanno sull'attributo classe:

$$H(E) = -\sum_{i=0}^{c-1} p_i \log_2 p_i$$

Dato che per ogni feature l'entropia calcolata sull'intero dataset è costante, allora le variabili possono essere classificate da IG semplicemente calcolando il secondo termine dell'equazione, quelle più predittive hanno un valore minimo per questo termine. Una volta stilata la classifica delle feature si possono scegliere le prime m (e.g 50% di tutte le originali), oppure si può determinare una soglia al di sotto della quale un attributo viene scartato.

L'Info Gain ha valori elevati su attributi che sono altamente informativi riguardo le istanze che si vogliono classificare. Se si considera un database di clienti di una compagnia telefonica allora il campo "CodiceFiscale" ha un valore elevato di IG visto che: individua univocamente gli utenti e permette una classificazione molto accurata dell'utenza. Purtroppo però l'attributo "Codice Fiscale" non permette alcuna generalizzazione. Per evitare il problema appena introdotto è possibile usare una misura della dispersione che si ottiene utilizzando un certo attributo: Gain Ratio introdotto in 2.2.4. Esistono molti altri criteri in letteratura[24] ma il principio di funzionamento è il medesimo.

Wrapper

Una delle caratteristiche principali delle tecniche *Filter* è il fatto di essere separate dall'algoritmo di induzione utilizzato dal classificatore. Questo non accade nell'approccio *Wrapper* in cui le performance di un classificatore guidano la ricerca nella selezione delle feature. Ad esempio il merito di un sottoinsieme di feature è dato dalla generalizzazione dell'accuracy che offre il classificatore utilizzando una Cross Validation sul training set. Se viene scelta la 10-Fold Cross Validation allora 10 classificatori sono costruiti e testati su ciascun sottoinsieme di feature. Questa tipologia di approccio ha un importante aspetto negativo: è computazionalmente costosa perchè, ad esempio, con un totale di m attributi lo spazio di ricerca ha taglia 2^m.

Tre stretegie popolari sono:

- Forward Selection: si inizia con un insieme vuoto di selezionate, si valutano tutte le opzioni includendo una feature alla volta, si seleziona la migliore tra queste e si considera una nuova situazione di partenza con un sottoinsieme di cardinalità uno a cui si prova ad aggiungerne un'altra.

- Backward Elimination: si parte utilizzando tutte le feature, si considera la nuova situazione eliminandone solamente una, si sceglie quale eliminazione porta al risultato migliore e si procede eliminandone via via una alla volta.

- Genetic Search: si utilizza un algoritmo genetico per cercare all'interno dei possibili sottoinsiemi di feature qual è quello ottimo. Nelle prime due strategie ci si ferma quando aggiungendo o togliendo variabili non si aumenta l'accuratezza. Entrambe sono strategie greedy che non garantiscono di trovare il sottoinsieme ottimo.

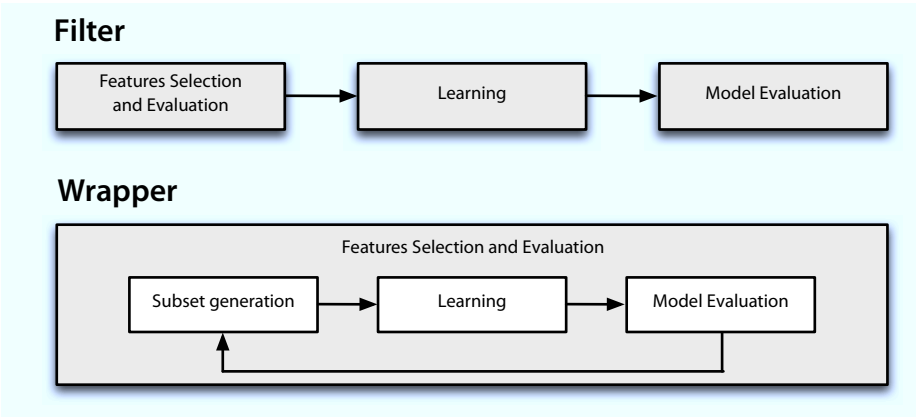


Figura 2.11: Confronto tra approccio Features Selection di tipo Filter e Wrapper

2.8.3 Sampling

L'utilizzo di tecniche di sampling prevede la modifica del dataset originale, con alcuni meccanismi, in modo da fornire una distribuzione bilanciata tra le classi. Sono molti gli studi [25][26][27] che hanno dimostrato come questi meccanismi migliorano le prestazioni di classificazione se comparate con quelle ottenute con dataset sbilanciati. Come precedentemente detto questo non implica che alcuni classificatori non possano apprendere bene anche da dataset sbilanciati.

Nel caso in cui si creino degli esempi artificiali (o sintetici) si parla di *oversampling*, se invece si eliminano dei record per bilanciare la distribuzione tra classi si tratta di *undersampling*.

Random Sampling

Il meccanismo del *random oversampling* segue esattamente la sua definizione: è un metodo non-euristico che cerca di bilanciare il dataset creando delle copie di esempi della classe di minoranza scelti in maniera random. Questa tecnica permette di aumentare la cardinalità della classe di minoranza a piacimento.

Mentre il random oversampling aggiunge dati al dataset di partenza, il *random undersampling* li rimuove. La scelta di quali record rimuovere viene fatta in maniera totalmente casuale, il numero delle eliminazioni dipende invece dal rapporto finale che si desidera ottenere tra le classi.

Molti autori sono concordi nel dire che il random oversampling può aumentare il rischio di overfitting visto che crea copie esatte della classe di minoranza. In questo modo un ipotetico classificatore può costruire delle regole che appaiono come accurate ma che in

realtà lo sono solamente perchè coprono esempi duplicati. Il problema legato al random undersampling è invece abbastanza ovvio visto che può portare ad eliminare informazioni utili al classificatore per discriminare in maniera accurata le classi.

Informed Sampling

Nel corso degli anni sono stati sviluppate molte tecniche con l'obiettivo di superare i limiti del random sampling: perdita di dati e overfitting. Di seguito vengono presentate le due tecniche utilizzate in questo lavoro.

SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) [28] è un metodo di oversampling che crea esempi artificiali della classe di minoranza basandosi sulla similarità nello spazio delle feature tra quelli esistenti.

Per ciascun record x_i appartenente alla classe di minoranza vengono presi in considerazione i *k-nearest neighbors* (vicini): sono definiti come i k record delle classe di minoranza la cui distanza euclidea con x_i presenta il minimo valore lungo le n dimensioni dello spazio delle feature. Per creare un esempio artificiale si sceglie in maniera casuale uno dei k vicini, dopodichè si esegue la differenza tra i due vettori di feature e si moltiplica il risultato per un valore random compreso tra 0 ed 1. Il nuovo esempio è la somma tra quanto ottenuto ed il vettore x_i.

$$x_{new} = x_i + (\widehat{x}_i - x_i) \cdot \delta_i$$

$\widehat{x}_i$ è uno dei k vicini di x_i e $\delta_i \in [0, 1]$ rappresenta il numero random. Operando in questo modo l'esempio creato si trova sulla linea che collega x_i con $\widehat{x}_i$ nello spazio delle feature. Il random oversampling ha il limite di creare una decision region, per la classe di minorananza, in maniera molto specifica; in modello quali i decision tree questo può portare a troppi split e di conseguenza all'overfitting. SMOTE risolve questi limiti cercando di generalizzare la regione di decisione della classe di minoranza.

Algoritmo 2.3 Creazione di un esempio sintetico con SMOTE

1) Trova i k vicini per ciascuna istanza appartenente alla classe di minoranza
2) Seleziona in maniera random il vicino x_j di x_i $(1 \leq j \leq k)$
3) Calcola la differenza tra i valori degli attributi dell'istanza x_i e il vicino x_j: $diff = x_j - x_i$
4) Genera un numero random δ (compreso tra 0 e 1)
5) Crea l'esempio sintetico: $x_{new} = x_i + diff \cdot \delta$

Gli step vengono ripetuti a seconda di quanti esempi si desidera creare. Può venire spontaneo pensare che maggiore sia il numero di esempi creati e maggiore sia l'accuratezza che si raggiunge visto che si hanno più record da cui apprendere. In realtà questo non è vero, vi è un limite al numero di istanze artificiali da creare oltre il quale si deteriorano le prestazioni poichè si vanno a creare esempi sempre più vicini tra loro, perciò simili.

SMOTE ha come limite quello di generalizzare in maniera "cieca" l'area della classe di minoranza, senza preoccuparsi di quella di maggioranza. Molto spesso quando si ha una situazione di elevato sbilanciamento non si hanno dei cluster ben definiti per le classi, può capitare che alcuni esempi della classe maggioranza invadano lo spazio di quella di minoranza. Andando a creare molti esempi artificiali si rischia una *overgeneralization*, ovvero si va ad intasare la regione della classe di maggioranza con esempi appartenenti ad un'altra classe rendendo difficile la distinzione durante la classificazione. Un esempio viene mostrato in Figura 2.12. Per questo motivo bisogna valutare bene la percentuale di record sintetici da creare e fermarsi prima di arrivare all'overgeneralization.

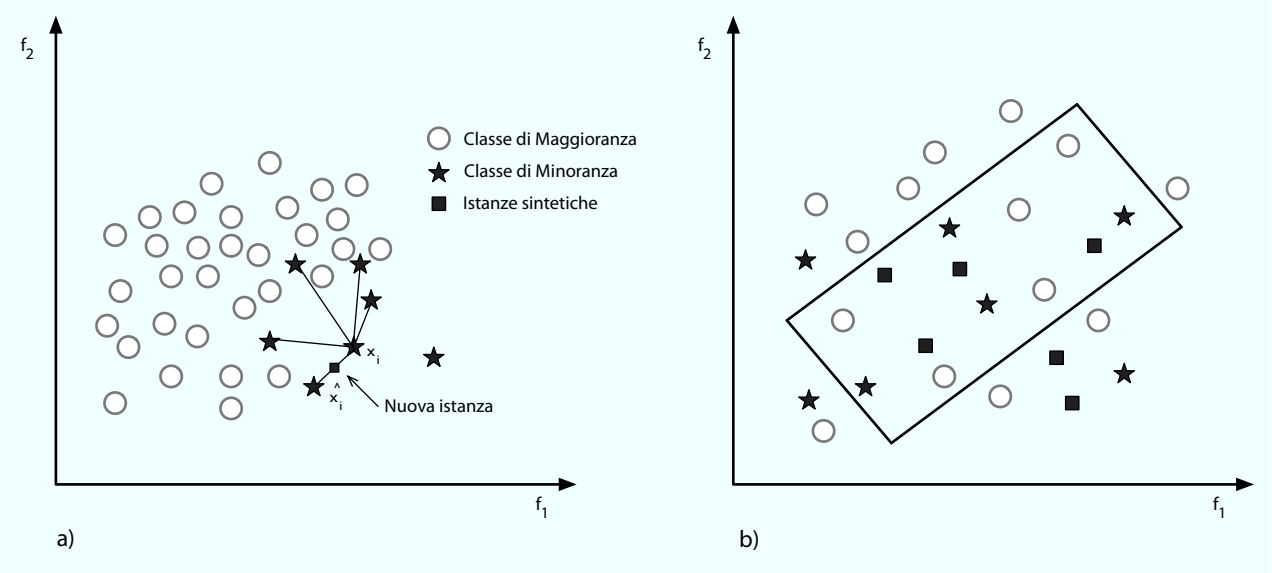


Figura 2.12: a) SMOTE b) Esempio di overgeneralization

Tomek Link

I *Tomek Links* possono essere definiti come una coppia di istanze a distanza minima appartenenti a classi diverse. Dati due esempi x_i e x_j con $x_i \in C^+$ e $x_j \in C^-$ si definisca con $d(x_i, x_j)$ la loro distanza. Una coppia (x_i, x_j) è chiamata Tomek Link se non esiste un record x_l tale che: $d(x_i, x_l) < d(x_i, x_j)$ oppure $d(x_j, x_l) < d(x_i, x_j)$. Perciò due

istanze formano un Tomek Link solamente in due casi: una di queste è frutto del rumore oppure entrambe si trovano vicino al confine che suddivide le classi (visto che non esiste nessun altro esempio lungo il segmento che li unisce) perciò si possono rimuovere. I Tomek Links vengono quindi spesso usati come tecnica di "pulizia" delle istanze che si trovano nella regione sbagliata (e.g istanze positive che si trovano circondate da istanze negative), molto spesso vengono utilizzati in accoppiata con SMOTE per cercare di risolvere il problema della sovrapposizione tra esempi spiegata prima (overgeneralization). Questo approccio può venire usato anche come metodo di undersampling qualora si decida di eliminare solamente i record, della coppia, che appartengono alla classe di maggioranza.

Capitolo 3

Identificazione dell'iperaldosteronismo primario

In questo capitolo viene descritto il problema medico oggetto di questa tesi. Nel Paragrafo 3.1 si parla delle caratteristiche dell'iperaldosteronismo, della prevalenza nella popolazione e delle strategie di screening utilizzate normalmente per la sua identificazione. Il Paragrafo 3.2 descrive invece le caratteristiche del lavoro fatto nel 1998 con il medesimo obiettivo di sviluppare un modello per la classificazione della malattia. L'ultimo paragrafo, il 3.3, contiene informazioni riguardo allo studio PAPY che ha permesso di stimare con buona precisione la prevalenza della patologia in Italia. Il dataset utilizzato per la realizzazione e la valutazione dei nuovi classificatori deriva proprio da questo studio effettuato nel 2006.

3.1 Definizione del problema medico

L'iperaldosteronismo primario (PA: Primary Aldosteronism) è stata descritto per la prima volta da Conn nel 1955 in una donna di 34 anni affetta da ipertensione arteriosa [29]. Si tratta di una comune causa, sebbene altamente sottodiagnosticata, di ipertensione arteriosa guaribile. Esistono due forme di ipertensione arteriosa: "essenziale"o primaria, della quale non si conosce la causa specifica, e secondaria dove invece è possibile stabilire la causa scatenante. L'iperaldosteronismo primario è una patologia in cui è presente un'inappropriata sovrapproduzione autonoma di aldosterone che non è sopprimibile attraverso l'uso di sodio. Esistono due forme (sottotipi) principali di PA: Iperplasia surrenalica bilaterale (BAH) o Iperaldosteronismo Idiopatico (IHA) e Adenoma Secernente Aldosterone (APA). Tra i malati di aldosteronismo primario si stima una frequenza relativa del 65% per quanto riguarda IHA, mentre gli APA arrivano a circa il 30%. Queste stime però

variano ampiamente a seconda del centro in cui sono realizzate e dall'uso del cateterismo surrenalico ai fini diagnostici.

Idiopatic HyperAldosteronism (IHA): in questa forma esiste una crescita eccessiva delle cellule secernenti aldosterone della ghiandola surrenalica, essa è solitamente diffusa e avviene in entrambe le ghiandole equamente. Il grado di sovrapproduzione di aldosterone è di solito relativamente moderato. È importante differenziare questa forma rispetto ad APA poichè IHA è trattato in maniera medica, solitamente con Spironolattone, un inibitore diretto di aldosterone. APA è invece trattato solitamente chirurgicamente, è fondamentale identificare perciò il sottotipo corretto.

Aldosterone Producing Adenoma (APA): è la causa più importante di iperaldosteronismo perchè è assai frequente ed è generalmente guaribile chirurgicamente, comunemente viene chiamata *sindrome di Conn*. Tipicamente in presenza di APA il livello di iperaldosteronismo è maggiore rispetto ad IHA. La sindrome di Conn è causata da un tumore benigno della ghiandola surrenale, il motivo per il quale questo adenoma si sviluppa non è del tutto chiaro. Una volta che la diagnosi è confermata viene effettuata una surrenectomia laparoscopica che permette di guarire l'iperaldosteronismo e di correggere l'ipertensione arteriosa.

Esistono altri sottotipi meno comuni di Iperaldosteronismo, uno dei modi più utilizzati per catalogarli è quello di distinguere le forme che possono essere guarite chirurgicamente da quelle che invece richiedono terapie farmacologiche.

Guaribili Chirurgicamente:

- Adenoma secernente aldosterone (Aldosterone Producing Adenoma, APA)
  - Unilaterale
  - Bilaterale
- Iperplasia surrenalica unilaterale (Primary Unilateral Adrenal Hyperplasia, PAH)
- Iperplasia surrenalica micronodulare unilaterale (Multinodular Unilateral Adrenocortical Hyperplasia, MUAN)
- Tumore ovarico aldosterone secernente
- APA o Iperplasia bilaterale (Bilateral Adrenal Hyperplasia, BAH) concomitante feocromocitomacarcinoma corticosurrenalico

Non Guaribili Chirurgicamente:

- Iperplasia Surrenalica Bilaterale (BAH)
- APA monolaterale con BAH
- Iperaldosteronismo familiare di tipo I (Familial Hyperaldosteronism type I, FH I) noto anche come iperaldosteronismo responsivo ai glucocorticoidi (o Glucocorticoid Remediable Aldosteronism, GRA)
- Iperaldosteronismo familiare di tipo II (Familial Hyperaldosteronism type II, FH II)
- Eccesso apparente di mineralcorticoidi (Apparent Mineralcorticoid Excess, AME)
  - Consumo cronico di liquirizia
  - Utilizzo di carbenoxolone (antiacido)

3.1.1 Conseguenze

A parità di aumento dei valori pressori, ed in presenza di un introito alimentare di sodio normale o alto, l'iperaldosteronismo primario induce un danno vascolare molto maggiore rispetto all'ipertensione essenziale poichè causa danno ossidativo al DNA, rimodellamento cardiovascolare, ipertrofia e fibrosi. Attraverso questi effetti deleteri esso altera il riempimento ventricolare sinistro e la funzione diastolica, induce fibrosi [1] nelle pareti del ventricolo sinistro, rallenta la conduzione atrio-ventricolare e prolunga l'intervallo PQ all'elettrocardiogramma facilitando l'insorgenza di blocco atrio-ventricolare, induce stiffening (irrigidimento) delle grandi arterie, rimodellamento di quelle di resistenza e microalbuminuria. Tutto ciò si traduce in un aumento del rischio di eventi cardiovascolari ed in particolare di fibrillazione atriale, ictus ischemico, emorragia cerebrale, edema polmonare "flash" ed infarto del miocardio.

3.1.2 Prevalenza

Poco prima della sua morte Conn riteneva che l'iperaldosteronismo primario avesse una prevalenza intorno al 7%, ma gli studi successivi hanno continuato a riportare tassi di prevalenza molto variabili (da 1.4% a 32% con una mediana di 8.8%). Questa ampia variabilità di stime derivava verosimilmente dal fatto che molti studi erano "viziati" da

[1] Deposizione di tessuto fibroso, solitamente come riparazione ad un infiammazione avvenuta per la quale non si può tornare allo stato del tessuto originale

fattori quali la natura di tipo retrospettivo[2], il coinvolgimento di casistiche selezionate e l'utilizzo di criteri molto variabili per la diagnosi. Per tutti questi motivi la prevalenza dell'iperaldosteronismo primario era praticamente ignota fino al 2004, anche se tra gli esperti era diffusa la convinzione che fosse molto maggiore rispetto a quanto comunemente ritenuto.

Nel 2006 è stato pubblicato il primo ampio studio prospettico: lo studio PAPY (Primary Aldosteronism Prevalence in hYpertensives) [30], progettato "ad hoc" per fornire dati "solidi" sulla reale prevalenza dell' iperaldosteronismo primario. Questo lavoro ha potuto dimostrare che l'iperaldosteronismo primario è la più frequente causa guaribile d'ipertensione su base endocrina poichè interessa oltre l' 11,2% dei pazienti ipertesi di prima diagnosi inviati a centri specializzati dell'ipertensione in Italia. Quasi la metà dei pazienti con iperaldosteronismo primario (4.8%) aveva una forma lateralizzata cioè chirurgicamente guaribile.

3.1.3 Strategia di screening

L'identificazione precoce delle forme chirurgicamente guaribili è molto importante poiché la completa guarigione chirurgica è tanto più plausibile quanto più tempestiva è la diagnosi. A questo proposito il test di screening più popolare per l'identificazione di PA è il rapporto tra i valori di aldosterone/renina (ARR). Esso rappresenta un metodo elementare ma il suo impiego richiede estrema attenzione a vari punti cruciali. Bisogna tenere presente che si tratta di un test che considera il rapporto tra due variabili, perciò con coppie di valori diversi si può ottenere il medesimo risultato. È quindi possibile avere una renina soppressa con un aumento dell'ARR nonostante si abbiano valori plasmatici di aldosterone (PAC) normali e quindi assenza di iperaldosteronismo. Per questo motivo il rapporto ARR va interpretato alla luce dei valori di PAC e del valore minimo di renina che il metodo di dosaggio impiegato è in grado di rilevare. A ciò si aggiunge il fatto che diversi farmaci influenzano il risultato di tale rapporto, inoltre non esiste un valore di cut-off universale ma piuttosto ciascun centro dovrebbe sceglierlo in modo tale da massimizzare l'accuratezza sui suoi pazienti. Per queste motivazioni si usano generlamente dei test di conferma, o esclusione, dopo l'ARR. Le linee guida dell'*Endocrine Society* suggeriscono come test di conferma: il carico orale di sodio, l'infusione di soluzione fisiologica salina, il carico salino+fludrocortisone e il test al captopril.

Come detto è necessario identificare il sottotipo di PA per ogni paziente, questo però non è possibile farlo con i test appena citati. Le linee guida prevedono dei test di imaginig (TC ad alta risoluzione) per l'identificazione. Tuttavia non sono da considerare affidabili per la discriminazione tra APA ed IHA. In realtà è stato dimostrato come le informazioni

[2]Sono studi condotti sulla base di documentazione raccolta in passato e, quindi, già esistente prima della decisione di iniziare lo studio; si tratta perciò di una ricerca d'archivio

che si deducono dalla TC possano essere fuorvianti, se considerate da sole, poiché in circa il 15-25% dei casi possono condurre ad una surrenectomia inutile ed escluderla nel 20% di coloro che potrebbero invece beneficiarne. Data l'inaffidabilità dei test di imaging la maggioranza degli esperti ritiene che l'AVS (cateterismo venoso surrenalico o Adrenal Vein Sampling) sia l'esame "principe" per la distinzione tra APA e IHA. L'AVS si basa sulla misurazione della concentrazione di aldosterone (PAC) e di cortisolo (PCC) nel sangue refluo delle vene surrenaliche del surrene destro e sinistro. L'AVS è un esame costoso, richiede elevate competenze tecniche ed è leggermente rischioso per il paziente (può esserci la rottura della vena surrenalica) per cui è importante sottoporvi i pazienti che presentano un'elevata probabilità di essere affetti da APA. L'obiettivo di questo lavoro è quello di fornire ai medici un modello matematico che possa aiutarli nella selezione dei pazienti sui quali eseguire l'AVS.

3.2 Modello del 1998

Nel 1998 è stato pubblicato un lavoro che ha cercato di risolvere lo stesso problema trattato in questo progetto, ovvero lo sviluppo di un modello che discrimini i pazienti affetti e non da APA. Il lavoro è stato pubblicato con il titolo: *Screening for primary aldosteronism with a logistic multivariate discriminant analysis* [1]. Utilizzando i dati provenienti da pazienti affetti da ipertensione si è sviluppato un modello logistico utilizzando una tecnica statistica chiamata Logistic Regression; questa permette di costruire una funzione che restituisce la probabilità che un paziente si affetto da APA sulla base di alcuni suoi valori biochimici.

Per lo sviluppo del modello logistico sono state impiegate le informazioni di 206 pazienti (32 affetti da APA) provenienti dalla Clinica Medica 4 di Padova. Ciascun paziente era identificato attraverso i valori di 20 variabili (riportate in Appendice C.1). A priori sono state scelte le variabili da utilizzare nel modello: solamente quelle che congiuntamente differivano tra pazienti con e senza APA. La valutazione del modello ottenuto è stato poi realizzata su due gruppi: il primo comprendeva 48 pazienti affetti da PA di cui 6 con APA, mentre il secondo raggruppava 320 ipertesi provenienti dalla clinica di Reggio Emilia, tra questi 11 risultavano affetti da adenoma di Conn e 8 da IHA.

Visto che tra le variabili scelte due erano tra loro linearmente dipendenti (sAldo e cAldo), e dato che l'analisi multivariata si basa sull'assunzione che queste non possono essere incluse nel medesimo modello, allora sono stati sviluppati due modelli ciascuno basato su tre variabili diverse.

Da quanto ottenuto appare che la maggior parte dei pazienti affetti da APA venivano correttamente identificati dal modello multivariato, nonostante ciò venivano riscontrati un numero di falsi positivi (principalmente pazienti affetti da IHA). I valori di sensibilità e

specificità dei due modelli sono riassunte nella Tabella 3.1. In sintesi i risultati mostrano che una strategia basata sull'analisi discrminativa multivariata, che usa informazioni derivanti da semplici test biochimici, può portare ad ottenere alta sensibilità, specificità ed accuratezza per l'identificazione dell'aldosteronismo primario in una popolazione selezionata di ipertesi che mostra un'elevata versomiglianza delle condizioni descritte. Essendo basata su test biochimici, che sono largamente disponibili nella maggior parte dei centri che si occupano della cura degli ipertesi, la strategia sviluppata appare essere realizzabile ed affrontabile, inoltre porta ad un aumento del valore diagnostico rispetto a test fondati sull'analisi di una sola variabile. Il modello sviluppato è sicuramente migliore, in termini diagnostici, dell'ARR che si ricorda essere una cruda analisi bivariata che fornisce valori uguali a partire da numeri molti diversi; la presenza di una funzione lineare multivariata, i cui coefficienti sono stati determinati con la Regressione Logistica, fa in modo che questo problema non sussista.

Coefficiente	Value	Standard Error	P-value
Modello A (sPRA, sAldo, serum K^+)			
`Intercept`	4.79	2.9105	-
`sPRA`	-2.5458	0.7366	< 0.00001
`sALDO`	0.0071	0.0022	< 0.00001
`K`$^+$	-1.6162	0.7104	$= 0.02$
Modello B (sPRA, cAldo, serum K^+)			
`Intercept`	2.5484	3.0237	-
`sPRA`	-2.8626	0.9473	< 0.00001
`sALDO`	0.0177	0.0037	< 0.00001
`K`$^+$	-1.4649	0.7610	$= 0.05$

	Sensibilità (%)	Specificità (%)	Accuratezza (%)
Padova (Retro)			
Modello A	100	68.4	73.0
Modello B	100	77.6	80.7
Padova (Prosp)			
Modello A	100	64.3	83.3
Modello B	100	80.9	87.5
R.Emilia (Retro)			
Modello A	100	69.3	70.3
Modello B	100	89.6	90.0

Tabella 3.1: Coefficienti dei Modelli sviluppati con Regressione Logistica e Risultati ottenuti

3.3 Studio PAPY

I dati utilizzati per lo sviluppo di questo progetto provengono dal citato studio PAPY. Si tratta di uno studio condotto con lo scopo di identificare la reale prevalenza dell'iperaldosteronismo primario nella popolazione degli ipertesi italiani. In questo lavoro, dopo un test di screening basato sul rapporto ARR, i pazienti venivano sottoposti ad un iter

diagnostico approfondito che permetteva non solo di stabilire la presenza o l'assenza della patologia, ma anche di identificarne il sottotipo. Per raggiungere tale scopo venne introdotto per la prima volta un set predefinito di criteri per diagnosticare con certezza l'APA denominato *"four-corners approach"* . Tale set era basato sulla dimostrazione d'ipersecrezione lateralizzata di aldosterone, sulla diagnosi istologica e sull'evidenza al follow-up della guarigione. Nei casi in cui non fosse possibile dimostrare una ipersecrezione lateralizzata di aldosterone veniva posta la diagnosi di iperaldosteronismo idiopatico (IHA).

Lo studio segue le raccomandazioni dello STARD (Statement for Reporting Studies of Diagnostic Accuracy). Per minimizzare la possibilità di un'alterazione del lavoro dovuto alla selezione dei pazienti, fu deciso a priori di registrare i pazienti con una nuova diagnosi (entro 6 mesi) di ipertensione indirizzati a centri specialistici dai medici di famiglia.

Il lavoro svolto ha dimostrato che l'11.2% dei pazienti ipertesi di prima diagnosi, inviati a centri specializzati per la cura dell'ipertensione, era affetta da iperaldosteronismo primario. Quasi la metà dei pazienti con PA (4.8% dei 1124 arruolati nello studio) aveva una forma lateralizzata, cioè chirurgicamente guaribile.

Il dataset realizzato sulla base dello studio PAPY è stato utilizzato come training set per lo sviluppo dei modelli presentati in questa tesi. Nel dataset sono presenti un totale di 1124 record suddivisi in quattro categorie:

1. **Non PA**: pazienti non affetti da PA (1001)
2. **AdenomaDX**: pazienti affetti da PA dovuto ad adenoma nel surrene destro (24)
3. **AdenomaSX**: pazienti affetti da PA dovuto ad adenoma nel surrene sinistro (28)
4. **Iperplasia Bilaterale**: pazienti affetti da PA dovuto ad iperplasia bilaterale (71)

Visto che per i medici è importante discriminare le persone con adenoma, poiché sono chirurgicamente guaribili, viene definito un problema di *classificazione binaria* dove le due classi sono APA e Non APA (racchiude al suo interno i pazienti non affetti da PA e quelli che presentano Iperplasia Bilaterale). Considerando tale problema nel training set il 4.6% dei record appartiene alla classe APA, mentre il restante 95.4% appartiene alla categoria Non APA. A partire dal dataset iniziale sono stati ricavati 30 attributi (29 + 1 target) che descrivono ciascun paziente (sono stati eliminati tutti gli attributi con un elevata percentuale di valori mancanti), alcuni numerici ed alcuni categorici, a questi si aggiunge una variabile che identifica il centro di provenienza del paziente. La lista completa delle variabili utilizzate è presente nell'Appendice C.2.

Capitolo 4

Risultati Raggiunti

In questo capitolo vengono presentati e discussi i risultati sperimentali ottenuti, e ne viene fatto un confronto. È descritto il procedimento che ha portato allo sviluppo dei vari modelli, sono inoltre spiegate nel dettaglio le caratteristiche di ciascuno di essi. Nei Paragrafi 4.1 e 4.2 si parla dei vari classificatori sviluppati (PET, SVM) ed analizzati (Modello '98). Sono esposti i vari test effettuati durante la fase di creazione dei PET e viene fatto un confronto tra le diverse tecniche di gestione dello sbilanciamento dei dati. Viene descritto il processo che ha portato allo sviluppo finale del modello SVM, il lavoro realizzato nel 1998 viene testato ed analizzato sul dataset PAPY. Si parla inoltre della combinazione tra i vari classificatori. Nel Paragrafo 4.4 vengono confrontati e valutati tra loro tutti i modelli per avere un giudizio finale generale su quanto ottenuto. L'analisi viene fatta sulla validazione Leave One Out Clinica (più pessimistica e realistica). In Appendice B è possibile vedere il confronto tra le curve ROC ottenute con essa e quelle dovute alla 10-Fold Cross Validation.

4.1 PET

Con l'obiettivo di provare ad usare un approccio diverso rispetto a quello utilizzato nel 1998 la scelta è ricaduta inizialmente sui PET. Come detto nel Capitolo 2 l'uso di questo strumento offre numerosi vantaggi, in particolare gli alberi permettono di realizzare un modello estremamente semplice e leggibile per i medici, inoltre consentono di identificare quali attributi meglio discriminano i pazienti malati da quelli non malati.

Per lo sviluppo dell'albero decisionale si è utilizzato il framework WEKA [31]: si tratta di un software open source scritto in Java, sviluppato dall'università neozelandese di Waikato e rilasciato sotto licenza GNU General Public License. In WEKA è presente una collezione di algoritmi utili per operazioni di data mining: classificazione, estrazione di regole, regressione, clustering, etc. Questo framework è molto usato nell'ambito di

lavori che riguardano la Knowledge Discovery, il suo successo è dovuto sia alla natura open source, che permette di modificarlo secondo le proprie esigenze, sia alla facilità di utilizzo. Il software contiene al suo interno J48, ovvero l'implementazione in Java del famoso algoritmo C4.5 descritto nei capitoli precedenti. WEKA ricorre al formato ARFF (Attribute-Relation File Format) per la rappresentazione dei dati: si tratta di un file di testo composto da un'intestazione che descrive il tipo di attributi (numerico o nominale) utilizzati e da un elenco di istanze usate per l'allenamento.

Il primo passo è stato quello di provare ad eseguire l'algoritmo C4.5 sull'insieme di dati a disposizione in modo da avere delle valutazioni preliminari dalle quali partire. Inizialmente la stima delle prestazioni è stata fatta utilizzando una 10-Fold Cross Validation di tipo Stratified.

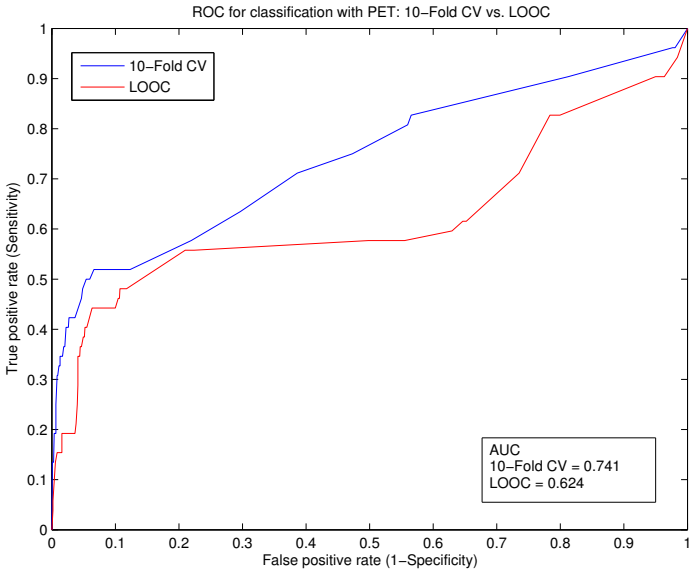


Figura 4.1: Confronto prestazioni: 10-Fold Cross Validation vs. LOOC

Come si può osservare dalla curva ROC 4.1 il valore di AUC che si ottiene è pari a 0.741. L'uso della Leave One Out Clinica (LOOC) invece fornisce prestazioni peggiori rispetto alla 10-Fold CV, ciò dimostra che inserendo record provenienti da cliniche diverse nella stessa fold si altera la valutazione. L'AUC che si ottiene in questo caso è pari a 0.624, perciò valori distanti dall'essere ottimali. Basti considerare che se si sceglie un valore di sensibilità ragionevole, ad esempio 80%, la relativa percentuale di specificità che si ottiene con questo modello è del solo 25%. Le varie coppie di specificità e sensibilità che verranno citate in seguito si ottengono al variare della soglia di classificazione scelta. Se

ad esempio si fissa una soglia del 10% (nel caso dei PET) significa che tutti i record con una probabilità maggiore al 10% di appartenere alla classe positiva vengono classificati come malati. Gli altri invece vengono etichettati come negativi. Con tale risultato di classificazione si hanno precisi valori di specificità e sensibilità. Variando la soglia di classificazione si vanno ad etichettare in modo differente i record e si ottengono perciò valori di sensibilità e specificità diversi. Quando più avanti si andrà ad affermare che si sceglie una determinata sensibilità, oppure specificità, significa che si sceglie una soglia che possa garantire tali valori. Si ricorda che la specificità indica la capacità di un modello (test) di dare un risultato negativo per una persona sana, la sensibilità invece rappresenta la capacità di individuare pazienti malati.

Le prestazioni non ottimali che arrivano con l'uso del semplice C4.5 sono dovute essenzialmente allo sbilanciamento dei record presenti nel dataset. Per cercare di limitare questo problema, ed ottenere prestazioni migliori, si è ricorso alle tecniche presentate nel Capitolo 2. D'ora in avanti le valutazioni verrano compiute utilizzando LOOC visto che fornisce una misura prestazionale più affidabile.

4.1.1 Tecniche di Sampling

Per mitigare la grande differenza tra il numero dei record appartenenti a classi diverse la prima scelta è ricaduta sulle tecniche di Sampling. WEKA offre molti filtri per processare i dati prima di iniziare la fase di classificazione, tra questi filtri è presente l'implementazione dell'algoritmo SMOTE che permette operazioni di oversampling. Sono disponibili anche delle tecniche che realizzano il random undersampling. SMOTE tra i suoi parametri di input permette di specificare la percentuale di nuovi record da creare rispetto al numero originale. Sono stati provati valori crescenti: 50%, 100%, 200%, etc. fino a 500%: tanto maggiore è la percentuale e tanto più si diminuisce la differenza di cardinalità tra le classi. Le prestazioni migliorano all'aumentare della percentuale fino ad arrivare al valore 200, da lì in avanti non si notano ulteriori miglioramenti; piuttosto con percentuali elevate, ad esempio 500%, le performance sono addirittura peggiori rispetto a quelle che si ottengono lavorando sul training set originale. Questo è indice del fatto che, con tali percentuali, si creano troppi esempi artificiali che vanno a sovraffollare lo spazio degli esempi rendendo difficile la distinzione tra classi. Si cade perciò nella situazione di overgeneralization citata nella Sezione 2.8.3. WEKA mette a disposizione il filtro *SpreadResample* che permette di impostare la proporzione tra le cardinalità delle due classi. Tale obiettivo viene raggiunto effettuando un random undersampling sulla classe di maggioranza. Si è partiti con rapporti molto bassi, del tipo 2:1, che garantiscono una sensibilità maggiore rispetto a quella ottenuta con il dataset originale, ma chiaramente anche una specificità peggiore. All'aumentare di tale rapporto il modello che si ottiene è sempre più in grado di riconoscere gli esempi negativi, questo avviene però leggermente a discapito di quelli positivi. Per cui

il rapporto tra sensibilità e specificità si è dimostrato essere inversamente proporzionale. Dopo vari tentativi con rapporti diversi si è arrivati alla conclusione che 6:1 offre i migliori risultati in termini di tradeoff tra sensibilità e specificità. In seguito si è poi scoperta l'esistenza di una versione modificata di WEKA [32] che è particolarmente adatta per gestire dataset sbilanciati. Essa contiene alcune tecniche presentate in [33], e sono implementati gli algoritmi *CNN*, *Tomek Link* e *SMOTE+Tomek Link*. L'uso dell'ultimo algoritmo menzionato permette di mitigare il problema dell'overgeneralization citato prima, inoltre offre risultati migliori rispetto a quanto si ottiene con il singolo SMOTE. La combinazione finale scelta per applicare il Resampling al training set consiste in: SMOTE (200%) +Tomek Link + SpreadSubsample (6:1). Per prima cosa viene eseguito l'algoritmo SMOTE sul training set originale, in questo modo si aumentano gli esempi positivi a disposizione di J48. Sul nuovo training set ottenuto vengono poi applicati in sequenza i due metodi di undersampling: Tomek Link e SpreadSubsample. Dalla Figura 4.3 si vede come con questa combinazione sia possibile arrivare a valori di sensibilità elevati, circa 80%, ma ciò accade a discapito della specificità che non arriva oltre il 70%. Aumentando la sensibilità ulteriormente, la specificità arriva invece a valori non accettabili di circa 40%.

È importante sottolineare che i metodi di sampling riguardano solamente il training set, cioè le istanze sintetiche create vengono utilizzate esclusivamente in fase di costruzione del modello, mentre per la validazione si ricorre ai record originali. Ciò significa che indipendentemente dalla percentuale di generazione di SMOTE e dal rapporto tra le classi che si sceglie, il numero totale dei record usati per la validazione è sempre 1124, quello di partenza. Il classificatore *FilteredClassifier* di WEKA garantisce che i filtri siano applicati solamente al training set e non al test set.

4.1.2 Cost-Sensitive Learning

Visto che si lavora in ambito medico è necessario cercare di ridurre al minimo il numero di falsi negativi, ovvero pazienti che sono affetti da APA ma che non vengono riconosciuti dal sistema. Mantenendo dei costi unitari per entrambe le tipologie di errore si ottiene un valore di False Negative Rate troppo elevato, perciò si è seguita la strada del Cost-Sensitive Learning. Inoltre considerando dei costi unitari si vanno ad equiparare due tipi di errore che nella realtà hanno un peso ben diverso.

È possibile fornire all'algoritmo J48 le penalità per le classificazioni errate attraverso una matrice dei costi, in questo modo l'albero viene creato tenendo conto dei diversi pesi dovuti agli errori commessi. Impostando un valore superiore per la classe positiva si ottiene un albero che cerca di limitare i falsi negativi, ovvero gli errori più pericolosi in ambito medico. Anche in questo caso sono stati testati dei valori crescenti per scoprire quale combinazione possa offrire il modello migliore in termini di prestazioni. Si è mantenuto un costo unitario per i falsi positivi mentre si è aumentato quello associato ai falsi negativi,

ovvero i pazienti affetti da APA non riconosciuti dal modello come tali. Scegliendo valori bassi, inferiori a 10, non si vedono grossi cambiamenti sulle prestazioni, mentre con valori maggiori si ottengono alberi che hanno una sensibilità crescente, a leggero discapito della specificità. Ad un aumento della sensibilità del 5% si è visto un abbassamento della specificità solamente del 2-3%. Quando si sceglie l'approccio Cost-Sensitive è necessario trovare il giusto compromesso riguardo numero e tipo di errori che si permette di compiere al modello. Se il costo per i falsi negativi è molto alto allora l'algoritmo sviluppa un albero che tende a classificare correttamente le istanze positive rispetto a quello negative, perciò con la diminuzione del numero di falsi negativi si assisterà inevitabilmente ad un aumento dei falsi positivi. Superando progressivamente il costo di 25 per i falsi negativi aumenta via via la percentuale di falsi positivi, abbassandolo invece la percentuale di sensibilità non arriva a valori soddisfacenti. Perciò dai risultati ottenuti si è scelto di costruire l'albero finale imponendo che il costo di un falso negativo sia 25 volte maggiore rispetto a quello di un falso positivo. Applicando questa tecnica si ottiene un valore di sensibilità attorno all' 80%, ma anche in questo caso la specificità è bassa, circa 73%. Dalla Figura 4.3 è comunque evidente il miglioramento rispetto all'applicazione del solo J48. Con percentuali di sensibilità più alte, ad esempio del 90%, la specificità raggiunge valori non ragionevoli attorno al 50%.

4.1.3 Feature Selection

Il modello sviluppato nel 1998 utilizza solo 4 variabili rispetto alle 20 disponibili per quel lavoro. Per questo motivo si è provato a diminuire la dimensione dello spazio delle variabili utilizzando la tecnica della Features Selection. Non si è invece considerata la Features Transformation dato che è fondamentale ottenere un modello finale i cui attributi abbiano dei valori interpretabili fisicamente.

Adottando l'approccio "Filtro" della Features Selection si è ricorso alla misurazione del Gain Ratio descritta nel Capitolo 2. Attraverso questo metodo le variabili vengono ordinate rispetto al valore di GR calcolato e solamente le prime m (parametro impostato dall'utente) vengono considerate in fase di costruzione dell'albero. Sono state fatte diverse prove selezionando rispettivamente: 20, 15, 10 e 5 variabili, le prestazioni ottimali sono arrivate con il valore 10. Utilizzando 15 o 20 variabili si ottengono dei risultati pressoché in linea con quelli raggiunti considerando l'insieme originale. Questo fa pensare che alcune delle variabili presenti nel dataset non sono di fatto discriminanti per l'identificazione della malattia. La Figura 4.2 riporta il Ranking che si ottiene applicando il Gain Ratio Evaluator sull'insieme delle variabili disponibili nel dataset.

```
Attribute Evaluator (supervised, Class (nominal): 30 Diagnosi):
        Gain Ratio feature evaluator
Ranked attributes:
0.2068     6 sK
0.1543    27 Aldo/PRA
0.1137    20 captoPRA
0.1092    24 cAldopmol/L
0.1076    23 captoAldo
0.0903    22 basAldopmol/L
0.0868    21 Aldobase
0.0698    19 PRAbase
0.0535     7 sNa
0.0481    29 PRAB02
Selected attributes: 6,27,20,24,23,22,21,19,7,29 : 10
```

Figura 4.2: Esempio di Feature Ranking in WEKA utilizzando Gain Ratio

Per quanto riguarda l'approccio "Wrapper" si è provata una ricerca del sottoinsieme ottimo utilizzando un algoritmo genetico [34]. L'insieme di variabili è risultato essere molto simile a quello ottenuto con GR, segno che effettivamente le variabili selezionate sono quelle che maggiormente riescono a fornire informazioni per l'identificazione delle due classi. Per la costruzione del modello finale è stato scelto GR come metodo di selezione delle feature.

Il solo utilizzo delle Feature Selection sul dataset PAPY non fornisce eccessivi miglioramenti. In Figura 4.3 è mostrata la ROC relativa alla classificazione basata sulle 10 variabili scelte dall'algoritmo Gain Ratio: come si vede questo approccio sembra portare ai miglioramenti più limitati. Tra Cost-Sensitive Learning e Resampling appare esserci una sostanziale parità per quanto riguarda l'aumento delle prestazioni che si ottiene rispetto all'uso del solo J48. Le curve infatti sono molte vicine tra loro lungo tutto l'area della ROC. Applicando contemporaneamente gli approcci descritti si arrivano ad ottenere i migliori risultati che si sono visti con la tecnica dei PET. Il modello finale, valutato con LOOC, ha una curva ROC con area 0.836. È interessante notare come in termini di AUC i singoli metodi di Resampling e Cost-Sensitive raggiungono pressochè i valori offerti dalla combinazione contemporanea delle tre tecniche, rispettivamente si ha 0.825 e 0.820; ciò farebbe pensare che la combinazione non porta ad eccessivi miglioramenti. Se si osserva però attentamente la Figura 4.3 si nota che l'uso contemporaneo delle tre tecniche fornisce una curva più vicina al punto (0,1) che indica la classificazione perfetta. Infatti se si confrontano i valori di specificità fissando quello di sensibilità a 80% si ottiene: 70% con Resampling, 73% con Cost Sensitive e ben 82% con l'utilizzo simultaneo degli approcci. Allo stesso modo se si fissa una specificità di 80% l'accoppiamento di tutte le tecniche

garantisce una sensibilità del 83% contro circa il 65% delle altre due tecniche. Il modello finale è quindi il risultato della combinazione delle tre tecniche presentate. L'albero viene costruito considerando solamente 10 variabili, vengono penalizzati maggiormente i falsi negativi, sono creati degli esempi artificiali per aumentare le istanze della classe positiva, e ne vengono eliminate alcune per abbassare la cardinalità di quella negativa.

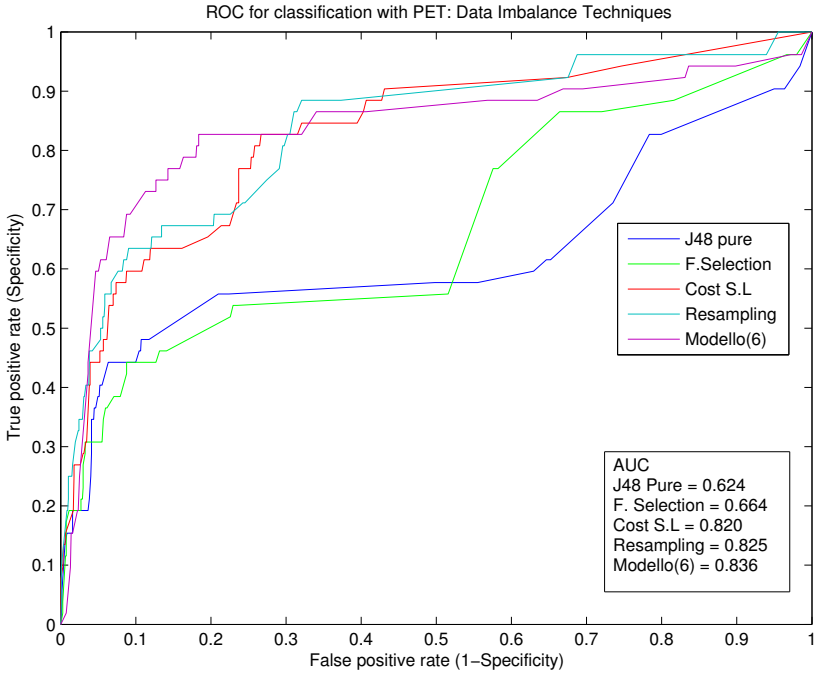


Figura 4.3: Confronto prestazioni tra tecniche diverse con LOOC

4.1.4 Pruning

Come detto nella Sezione 2.2.7, le tecniche di pruning portano ad una riduzione della precisione sulla stima delle probabilità. Per questo motivo gli alberi creati sono *unpruned*, cioè la loro dimensione non è stata ridotta da nessuna operazione di post-pruning. Così facendo si ottengono dei modelli più complessi e con un numero di nodi interni maggiore ma che garantiscono stime di probabilità più precise.

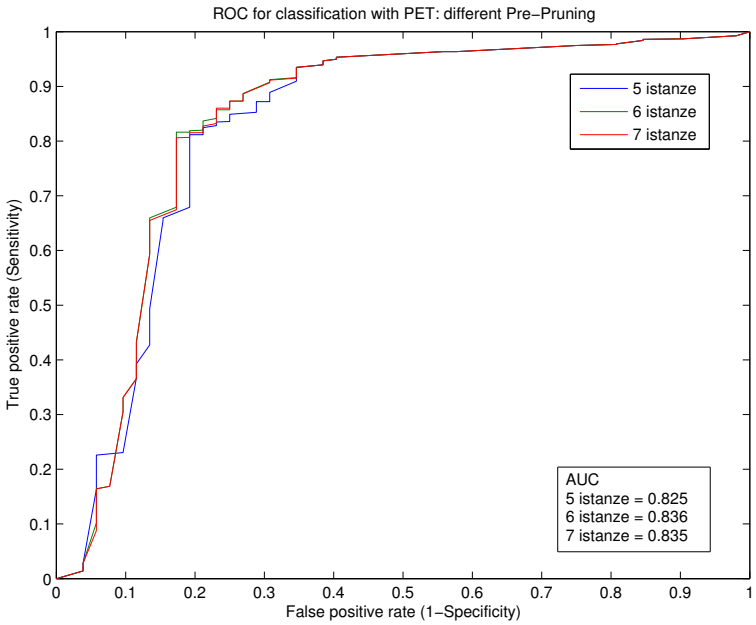
In fase di costruzione è possibile specificare il numero minimo di istanze che ciascuna foglia deve contenere, si tratta dell'implementazione della *StoppingCondition(E,A,C)* presente nell'algoritmo di Hunt. Selezionando questo valore minimo si effettua comunque un'operazione di pruning, più precisamente di pre-pruning poiché si arresta lo sviluppo

dell'albero quando viene raggiunta tale soglia. Di default il valore scelto da WEKA è 2, ciò porta ad avere degli alberi estremamente profondi: si sono perciò provati valori maggiori per cercare di limitare la taglia del modello, rispettivamente 5,6 e 7. Le curve ROC ottenute con parametri diversi sono confrontate nella Figura 4.4. Si può notare che le prestazioni con 6 e 7 sono molti simili, mentre con il valore 5 si ottiene qualcosa di leggermente inferiore. Si è deciso di scegliere come modello finale l'albero che ha un numero minimo di record in ciascuna foglia pari a 6, esso fornisce una curva ROC con area di 0.836 rispetto a 0.825 del modello ottenuto scegliendo come valore minimo 5. D'ora in avanti si identificarà tale modello, costruito con il valore minimo di istanze nelle foglie pari a 6, con Modello(6), mentre Modello(5) indicherà una scelta pari a 5. La preferenza nei confronti del Modello(6) è dovuta alla migliore AUC ed alla maggiore vicinanza, rispetto alle altre configurazioni, al punto (0,1) nello spazio ROC. Le varie coppie di sensibilità-specificità si possono vedere in Figura 4.4, esse si riferiscono alla validazione LOOC. In Appendice A sono mostrati gli alberi completi che si sono ottenuti. Si nota che il Modello(6) rispetto al Modello(5) ha un numero di test inferiore, la dimensione[1] infatti è minore: 21 rispetto a 33. L'altezza[2] di entrambi gli alberi è invece 8. Si può quindi concludere che il Modello(6) appare meno complesso ed articolato rispetto al Modello(5). In entrambi i casi le probabilità associate alle foglie sono state calcolate con la correzione di Laplace.

Osservando la Figura 4.4 è evidente il miglioramento che si ottiene confrontando le prestazioni dell'albero finale con quelle del modello sviluppato dal solo algoritmo J48, in termini di AUC si passa dal valore 0.624 a 0.836. Dalla Tabella 4.4 appare che per la coppia sensibilità-specificità si riescono ad avere contemporaneamente valori attorno all' 80%, cercando invece una soglia che porti ad una sensibilità molto elevata si ottiene un brusco calo dei relativi valori di specificità. È lampante il progresso ottenuto rispetto ai primi test in cui si raggiungeva una specificità del solo 25% per valori di sensibilità dell' 80%.

[1] Numero totale dei nodi di un albero

[2] Massima distanza di una foglia dalla radice dell'albero



Modello(6)

Specificità	Sensibilità
96.22%	39.78%
95.88%	60.45%
90.80%	70.76%
82.59%	80.76%
66.35%	86.45%
30.5%	90.38%
17.32%	94.23%

Figura 4.4: Confronto prestazioni con LOOC per diversi valori di pre-pruning

4.2 SVM

I PET forniscono un modello estremamente facile da comprendere ed offrono una serie di informazioni notevoli. Presentano però dei limiti in termini di prestazioni, sopratutto in presenza di un training set sbilanciato. Per questo si è provato a creare un nuovo modello utilizzando una tecnica di learning diversa: Support Vector Machine (SVM).

Per lo sviluppo del modello attraverso SVM si è adoperata la libreria open source LibSVM [35], in particolare la sua implementazione in Matlab. Rispetto a WEKA l'uso di Matlab consente di sviluppare in maniera più personale il modello, si possono infatti creare ed utilizzare funzioni in base alle necessità che il problema richiede. Ad esempio in questo lavoro si è utilizzata un'implementazione open source dell'algoritmo EM. Qualora in futuro si decidesse di provare ad applicare nuovi algoritmi o nuovi approcci per la classificazione potranno essere aggiunti al codice sviluppato finora senza problemi.

SVM a differenza dei Decision Tree non gestisce i missing values, perciò è necessario usare un algoritmo che rimpiazzi i valori mancanti. Inizialmente si è fatto ricorso alla funzione *KnnImpute*, presente in Matlab, che sostituisce i valori mancanti con la media pesata dei valori presenti nei k-nearest neighbours; i pesi vengono assegnati sulla base delle distanze con il record vicino. In un secondo momento si è invece scelto l'algoritmo EM

presentato nel Capitolo 2. Come accennato, la sua implementazione open source in Matlab è disponibile online e garantisce risultati migliori rispetto al *KnnImpute*. È importante sottolineare che il riempimento dei missing values va fatto utilizzando solamente il training set. Ciò significa che in fase di validazione (sia essa LOOC o Cross Validation) i parametri utilizzati da EM per riempire i missing values sono calcolati considerando solamente i record del training set. Le istanze del test set vengono invece completate utilizzando sia il training set che lo stesso test set, lo pseudocodice che mostra come ciò avviene è disponibile in Figura 4.5.

```
TrainFilled = EM(Train);
for j=1:size(Test)
 x999=[TrainFilled; Test(j,:)];
 P=EM(x999);
 TestFilled(1,:)=P(size(P),:);
end
```

Figura 4.5: Pseudocodice per il riempimento dei missing values del test set

Prima di allenare il modello è importante normalizzare gli attributi [36]. Il primo vantaggio è quello di evitare che gli attributi con un range numerico grande possano dominare quelli con range minori, inoltre si evitano difficoltà in fase di calcolo. Per normalizzare si utilizza il metodo *Min-Max*: per ogni attributo si considera il valore massimo e minimo presente nel dataset, a ciascun valore originale viene poi sottratto il minimo e viene diviso per la differenza tra massimo e minimo. In questo modo si ottengono attributi che hanno range compreso tra 0 e 1.

Inizialmente si sono provati i due kernel che maggiormente vengono utilizzati: lineare e gaussiano (RBF). Per valutare i modelli si è considerata ancora una volta la validazione Leave One Out Clinica. Quando si usano questi Kernel le differenze in termini di prestazioni arrivano dalle scelte che vengono fatte rispetto ai parametri C e γ; per cui l'obiettivo iniziale è stato quello di determinare quali valori offrivano le performance migliori. Si ricorda che il parametro C indica la penalità che viene assegnata in caso di classificazione errata da parte del modello, mentre γ rappresenta invece l'esponente della funzione esponenziale del Kernel. Un valore elevato di γ porta ad ottenere margini "più irregolari" che si adattano bene agli esempi, al contrario bassi valori evitano l'overfitting. È stata effettuata una ricerca dei parametri ottimali utilizzando una *GridSearch* (vedi Sezione SVM del Capitolo 2) con diverse coppie di valori C, γ. Da questa ricerca è uscito che il Kernel RBF garantisce i risultati migliori utilizzando come parametri $C = 1$ e $\gamma = 0.03125$, la ROC

che si crea da tale combinazione, con LOOC validation, ha un'area di 0.854. In questo caso ad una sensibilità dell' 83% corrisponde una specificità dell' 80%.

Già da questi primi risultati si è visto come le SVM possano offrire un modello migliore rispetto ai PET, infatti utilizzando un semplice Kernel RBF si possono raggiungere, e superare, le prestazioni fornite dal miglior albero realizzato. Per cercare di aumentare ulteriormente le perfomance si è implementato il metodo *Random Subspace* che permette di creare tante SVM "ridotte", allenate su un dataset che ha uno spazio delle feature minore rispetto all'originale, le cui uscite vengono combinate per fornire la classificazione finale. Lo score che viene assegnato a ciascuna istanza del test set è il risultato della media dei vari score che ogni singola SVM calcola. Come descritto in 2.5, nel Random Subspace sono fondamentali due parametri: *Set* e *S,* ovvero il numero di classificatori (SVM) che si desidera creare per assemblare il modello finale ed il numero delle feature del dataset originale che si selezionano per allenare ciascuna SVM.

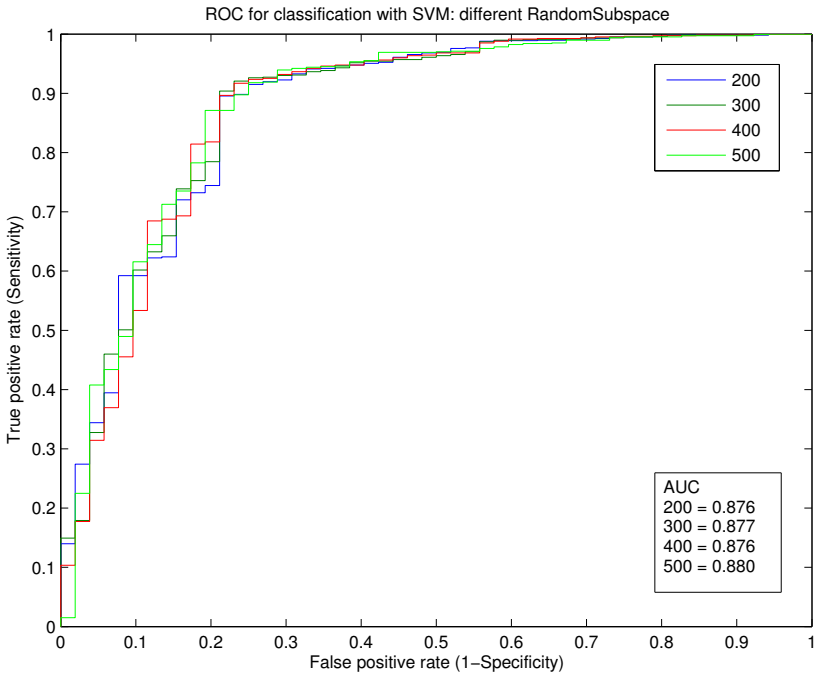


Figura 4.6: Confronto prestazioni tra RandomSubspace differenti a parità di parametri C=1 $\gamma = 10$

L'esecuzione del Random Subspace è computazionalmente pesante, sopratutto quando si utilizza un valore di *Set* elevato, per questo motivo sono state fatte delle prove suc-

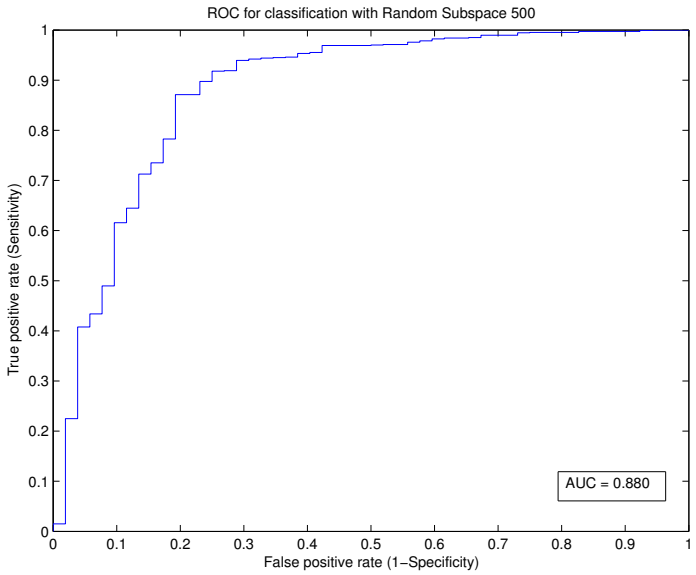
cessive considerando via via valori crescenti per tale parametro. Si è partiti con 200 e si è aumentato ad intervalli di 100 fino ad arrivare a 500. La comparazione tra i risultati ottenuti scegliendo valori di *Set* diversi si può vedere nella curva ROC 4.6. Come spiegato nella Sezione 4.2 le SVM restituiscono, per ogni istanza di test, uno score che rappresenta la distanza rispetto all'iperpiano di separazione. Le curve ROC per l'analisi delle prestazioni ottenute con le SVM sono state realizzate utilizzando questi score restituiti in output, il procedimento che porta alla loro creazione è il medesimo descritto in 2.6.2. In questo caso invece che utilizzare delle probabilità si è ricorso a dei punteggi positivi e negativi. Qualora, in futuro, si decida di usare un metodo per la stima delle probabilità a partire dagli score SVM, ad esempio quello di Platt descritto in 2.3.4, si possono ottenere delle nuove curve ROC sulla base dei nuovi valori calcolati. Dato che le probabilità sono stimate come una funzione degli score allora le due curve ROC saranno coincidenti o eventualmente leggermente diverse. Convertendo le soglie di score, associate alle varie coppie specificità-sensibilità, in soglie di probabilità è possibile mantenere valide le analisi effettuate sulle prestazioni.

Si osservi come le curve sono molto vicine tra di loro, in alcuni punti vi è addirittura una sovrapposizione. In termini di AUC primeggia il valore 500 con 0.880 anche se le altre scelte non sono molto distanti: ad esempio con 300 SVM si ottiene un'area di 0.877. La decisione finale è ricaduta su 500 poichè è quello che offre il valore più alto di sensibilità, rispetto alle altre configurazioni, a parità di specificità. Se per esempio si fissa la specificità ad 80% si può arrivare ad una sensibilità di circa 87% con il parametro 500, con 400 si arriva solo all'83%, con 300 si arriva sotto l'80% e così via. Il modello che scaturisce da tale configurazione prende il nome di Random Subspace 500, o abbreviato RS 500.

Le variabili che vengono prese in considerazione per creare i vari training set sono scelte in maniera random, perciò eseguendo due volte consecutive lo stesso codice di RSM (RandomSubspaceMethod) si possono ottenere modelli finali leggermente diversi. Ad esempio, ipotizzando di allenare considerando solo 5 variabili, alla prima esecuzione di RSM la SVM-1 (la prima SVM dell'insieme formato dalle 500 SVM) può essere allenata su $F_{SVM-1} = \{A_3, A_9, A_{17}, A_{19}, A_{23}\}$, mentre in una successiva esecuzione si può ottenere $F_{SVM-1} = \{A_1, A_8, A_{10}, A_{25}, A_{28}\}$. Visto che il numero di SVM è molto elevato le prestazioni che si ottengono con esecuzioni successive sono simili, anche se le singole SVM-i non sono esattamente identiche. La differenza, in termini di AUC, tra esecuzioni successive di Random Subspace 500 non è significativa, si parla di un valore attorno a 0.006. Perciò non si è ritenuto necessario cercare di eliminare questa differenza tra esecuzioni consecutive dello stesso algoritmo.

Il modello definitivo, per quanto riguarda l'approccio SVM/RandomSubspace, è stato quindi creato utilizzando un Kernel RBF (gaussiano) dove il parametro γ ha valore unitario, per quanto riguarda C si è scelto invece il valore 10. Si è notato che valori superiori di C non cambiano il risultato della classificazione. Infine si è deciso di allenare ciascuna

singola SVM su un training set che contiene un terzo delle variabili presenti nel PAPY. Scegliendo un insieme di feature minore le SVM possono non avere sufficienti informazioni per la classificazione. Essendo la selezione degli attributi casuale, può capitare infatti che molte SVM lavorino su delle variabili non significative a fini della classificazione. Usando metà degli attributi non si migliorano di fatto i risultati. Nella Figura 4.7 si può osservare la curva ROC del modello finale sviluppato, l'AUC è pari a 0.880.



RS 500

Specificità	Sensibilità
97.0%	40.3%
91.2%	61.23%
87.10%	71.45%
81.73%	80.98%
81.52%	85.91%
73.51%	90.86%
62.64%	95.38%

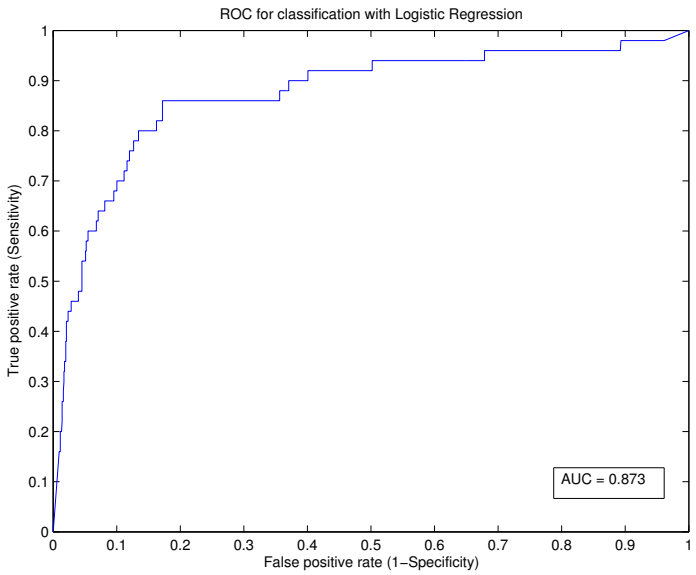
Figura 4.7: Prestazioni RandomSubspace 500

Può venire spontaneo chiedersi come mai non siano state utilizzate le varie tecniche presentate nel Capitolo 2 per combattere lo sbilanciamento dei dati. Il problema è che in una situazione di dataset sbilanciato, in cui si hanno così pochi esempi positivi, può essere rischioso cercare di aggiungere molti parametri che sono basati su di essi. Il pericolo è quello di creare una situazione di *overtraining* nella quale il modello diventa troppo complesso per poter poi generalizzare bene su nuovi dati. Cercare di compiere delle scelte specifiche fondate esclusivamente su un numero di esempi positivi limitati può portare ad un modello che non generalizza. Qualora questo venga utilizzato successivamente su un insieme di record diversi può classificare in maniera errata proprio a causa di queste scelte. Le prestazioni di partenza della semplice SVM (considerando il solo uso di un Kernel Lineare o Gaussiano) sono di gran lunga migliori rispetto a quelle ottenute con il semplice J48 (basti pensare che utilizzando la Leave One Out Clinica si ha un AUC di circa 0.84 contro 0.62). Questo fa pensare che il dataset, con il suo rapporto di 20:1 tra

classe di maggioranza e minoranza, possa apparire meno "sbilanciato" e difficile da gestire per la SVM rispetto ai PET. Visto quindi il buon punto di partenza, usando le SVM si è preferito non utilizzare le tecniche di Cost-Sensitive e Feature Selection; il Random Subspace mira ad essere un'alternativa a quest'ultima. Per quanto riguarda l'oversampling non è invece garantito che possa portare a dei miglioramenti se usato prima di SVM, infatti, idealmente, creando degli esempi artificiali si potrebbe rendere più difficile l'identificazione dell'iperpiano di separazione tra le due classi sovraffollando la zona di separazione. Al momento della realizzazione di questo progetto non è stata trovata un'implementazione open source per Matlab di qualche tecnica di oversampling. Non si è potuto quindi valutare se, con questo particolare problema, si possano ottenere dei miglioramenti nei risultati finali. Vale comunque la pena di provare in futuro, perciò tra gli Sviluppi Futuri c'è l'idea di implementare ed applicare qualche tecnica di resampling, evenutalmente più moderna rispetto a quelle considerate finora.

4.3 Ensemble con Modello '98

Per poter conoscere se gli obiettivi che ci si era posti in questa tesi sono stati raggiunti è necessario confrontare i classificatori sviluppati con quello realizzato nel 1998.Visto che i dati di quel periodo sono diversi rispetto a quelli attuali non è possibile prendere in considerazione i valori di specificità e sensibilità presenti nell'articolo. Avendo a disposizione i coefficienti da [1] si è facilmente implementato il modello B, quello che secondo la pubblicazione offre risultati migliori, e lo si è testato sul PAPY. Non si è riusciti ad applicare il modello a tutti i record dato che alcuni di essi non presentano dei valori per le quattro variabili sulle quali viene stabilità la probabilità. I record con missing values non sono stati trattati con nessun algoritmo specifico di gestione dei missing values visto che nel 1998 questo non accadde, così facendo si è voluta ottenere una validazione più veritiera possibile. Tali record non sono stati perciò considerati per questa analisi, in totale si sono perciò utilizzati 1108 record su 1124. L'AUC della curva ROC che si riferisce a tale validazione ha un valore di 0.873. La Tabella 4.8 dimostra che i valori di sensibilità e specificità citati nell'articolo non sono confermati, infatti in questo caso non si raggiunge praticamente mai la perfezione nell'identificazione dei pazienti positivi (True Positive Rate). Il valore più alto di sensibilità, circa 97% viene accompagnato da una specificità bassissima attorno al 10%. Risultati più ragionevoli arrivano abbassando la sensibilità a 86% con il relativo valore di specificità di 83%. In [1] entrambi i modelli sviluppati avevano una sensibilità del 100%, la specificità era invece rispettivamente del 69.3% e dell'86.9%. Tali percentuali non vengono mai raggiunte con il dataset PAPY.



Modello '98

Specificità	Sensibilità
98.0%	40.00%
95.3%	60.22%
88.21%	71.88%
87.73%	80.18%
80.17%	86.00%
72.11%	89.32%
50.42%	94.23%

Figura 4.8: Prestazioni modello '98 su dataset PAPY

4.3.1 Sum Rule

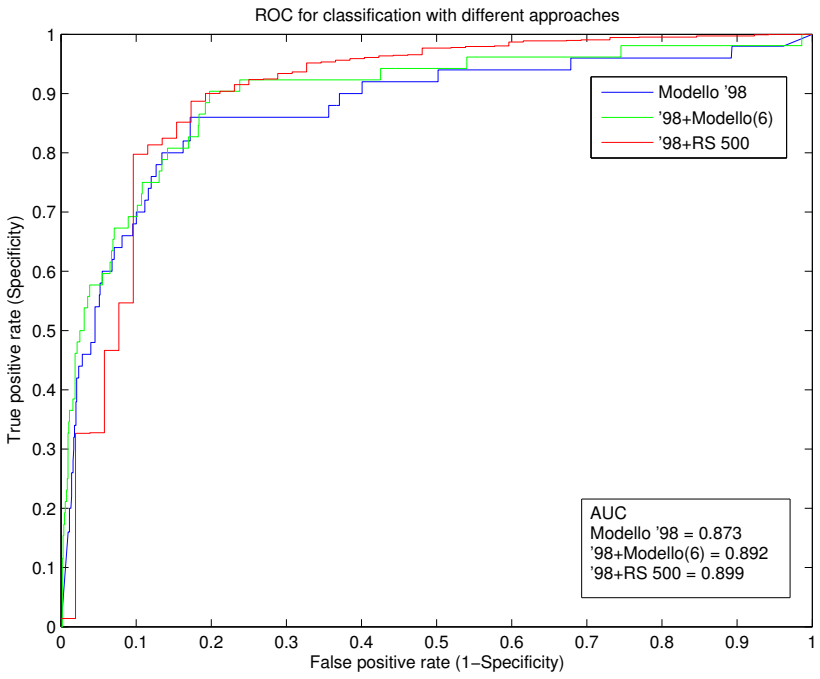
Analizzando le prestazioni del modello sviluppato nel 1998 si è osservato che sono molto buone, di fatto possiamo dire che in generale equivalgono quelle di RS 500 (una comparazione più approfondita viene fatta nella sezione successiva). Per questo motivo è nata l'idea di provare a combinare i classificatori sviluppati in questo lavoro con quanto realizzato attraverso la Regressione Logistica. Esistono diversi metodi disponibili in letteratura che permettono di aggregare più classificatori [37]. Tra questi vi è il metodo *Sum Rule* che combina le uscite di N classificatori facendone la media. Sia $f_i^j(x)$ l'uscita del classificatore j per il pattern x rispetto alla classe i, allora la sum rule prevede:

$$f_i^{s.r}(x) = \frac{1}{N} \sum_{j=1}^{N} f_i^j(x)$$

Nel caso in cui l'output di ciascun classificatore sia una probabilità, perciò un valore compreso tra 0 e 1, basta semplicemente applicare la formula. Qualora si combinano modelli che forniscono in uscita score con range diversi è prassi normalizzarli in modo che abbiano media nulla e varianza unitaria. Un metodo comune è lo *Z-Score*: $score_{norm} = \frac{(score-\mu)}{\sigma}$ dove μ e σ rappresentano rispettivamente la media e la deviazione standard di tutti gli score.

La combinazione (Ensemble) dei 2 classificatori coincide con la media degli output di entrambi i modelli, perciò quello del '98 deve poter essere applicato a tutti, anche ad istanze con missing values. Non è possibile avere degli esempi di test che sono classificati solamente dal PET, o da SVM, poichè la media risulterebbe corrotta. Per risolvere tale problema si è ricorso ancora una volta ad EM, in questo modo, a differenza della fase di valutazione del solo modello '98, descritta nella sezione precedente, si sono potuti considerare tutti i 1124 esempi del PAPY. Il procedimento seguito per il riempimento dei missing values è il medesimo presentato in 4.5. La combinazione tra PET e Logistic Regression è data dalla media delle probabilità che ciascun modello restituisce per ogni singolo record del test set, non è stato necessario alcun tipo di normalizzazione. Per la combinazione con il modello del '98 si è utilizzato il PET che ha fornito i migliori risultati, come spiegato nella Sezione 4.1 questo è il Modello(6). Da adesso in avanti l'ensemble tra PET e modello del '98 verrà identificato come '98+Modello(6). Con tale aggregazione si ottiene una curva ROC con AUC pari a 0.892.

Come detto in 4.2 l'output del SVM rappresenta la distanza dell'esempio di test dall'iperpiano di separazione. Non ha senso fare la media tra questa distanza e la probabilità restituita dal modello del '98, si tratta di misure con significati e range diversi. Per poter combinare gli output si è utilizzato lo Z-Score, in questo modo sia le probabilità che le distanze sono trasformate in un insieme di valori a media nulla e varianza unitaria. Dopo questa trasformazione degli output è possibile combinare tra loro i due modelli secondo il metodo Sum Rule. Anche in questa caso per la combinazione si è utilizzato il miglior modello sviluppato con la tecnica delle SVM, ovvero il Random Subspace 500; il modello risultante prende il nome di '98+RS 500. La sua curva ROC ha un valore, in termini di AUC, di 0.899. Come era prevedibile immaginarsi '98+RS 500 è migliore rispetto a '98+Modello(6), sia in termini di AUC che, quasi sempre, in termini di sensibilità-specificità come dimostrano i valori della Tabella 4.9. Considerando dei valori di specificità compresi tra 100% e 90% l'ensemble '98+Modello(6), a parità di specificità, garantisce una sensibilità sempre maggiore rispetto a '98+RS 500. Prendendo in analisi i valori della Tabella 4.9 si vede che con una specificità attorno al 95% '98+Modello(6) prevede una sensibilità vicina al 60% contro il 40% di '98+RS 500. Le cose si invertono quando la specificità inizia a scendere, a partire dal valore di 85% in poi '98+RS 500 permette una sensibilità sempre superiore o eventualmente uguale rispetto al corrispettivo ensemble generato con i PET. È inoltre immediato osservare come la curva blu, '98+RS 500, sia più proiettata verso il punto di classificazione perfetta (0,1) rispetto a quella rossa, '98+Modello(6). Spingendo la sensibilità a valori elevati, quindi in una situazione in cui il classificatore ha un'alta probabilità di riconoscere i pazienti malati, ad esempio attorno al 95%, si nota una differenza di addirittura 10 punti percentuali a livello di specificità. In generale, vista la maggiore AUC ed i confronti tra le coppie di valori specificità-sensibilità si può affermare che '98+RS 500 è leggermente superiore rispetto a '98+Modello(6).



'98 + Modello(6)

Specificità	Sensibilità
99.59%	40.38%
94.29%	61.56%
90.28%	71.15%
84.02%	80.77%
82.88%	84.62%
80.13%	90.38%
58.46%	94.23%

'98 + RS 500

Specificità	Sensibilità
95.34%	40.36%
89.45%	61.11%
89.11%	70.71%
89.11%	80.11%
85.33%	85.07%
81.49%	90.31%
68.79%	95.12%

Figura 4.9: Confrontro prestazioni tra gli ensemble sviluppati

4.4 Confronto tra classificatori

In questa sezione vengono messi a confronto tutti i classificatori sviluppati. Le prestazioni vengono comparate sulla base delle curve ROC, sia in termini di AUC e sia nel confronto tra le coppie specificità-sensibilità quando uno di questi due parametri viene fissato. Il primo paragone avviene tra il modello del '98 ed i due modelli sviluppati nella fase iniziale del progetto: Modello(6) e RS 500. La Figura 4.10 mostra le curve ROC ottenute da questi tre modelli. In termini di AUC RS 500 ha un valore maggiore rispetto agli altri due: 0.880 contro 0.873 del modello '98 e 0.836 del Modello(6). È importante non soffermarsi esclusivamente sul valori di AUC, sopratutto considerando che le differenze non sono troppo elevate. Analizzando in dettaglio la Figura 4.10 e le Tabelle 4.13 si nota che per valori di specificità compresi tra 100% e 80% il Modello(6) e quello del '98 hanno dei valori di sensibilità maggiori rispetto a quelli del RS 500. Ad esempio con una specificità pari a 90% con RS 500 si arriva ad una sensibilità del 61%, mentre con gli altri due modelli si ottiene un valore attorno al 70%. In questo range di specificità (tra 100% e 90%) il Model-

lo(6) ed il modello '98 garantiscono prestazioni molto simili, con una leggera superiorità dell'ultimo. Si parla comunque di qualche punto percentuale maggiore, niente di estremamente significativo. Quando si fissa un valore di specificità più basso, a partire dall'80% fino ad arrivare anche allo 0%, RS 500 inizia ad offrire performance migliori rispetto agli altri due classificatori. Fissando la specificità all'80% il Modello(6) ottiene una sensibilità dell'82% circa, gli altri due modelli si attestano attorno all'86%. Il divario aumenta se si sceglie come soglia di specificità 75%, infatti RS 500 in questo caso arriva a sensibilità del 92% contro il solo 86% del modello '98 e l'82% del Modello(6). Ragionando in maniera opposta, perciò fissando un elevato valore di sensibilità, 95%, RS 500 primeggia con 62% di specificità, rispetto al modello '98 che si ferma al 50% e Modello(6) che scende fino al 17%. È plausibile affermare che le prestazioni, in termini di sensibilità, del modello del '98 e quelle del Modello(6) si equivalgono quando la specificità assume valori molto elevati, superiori al 90%, in caso contrario primeggia il modello RS 500.

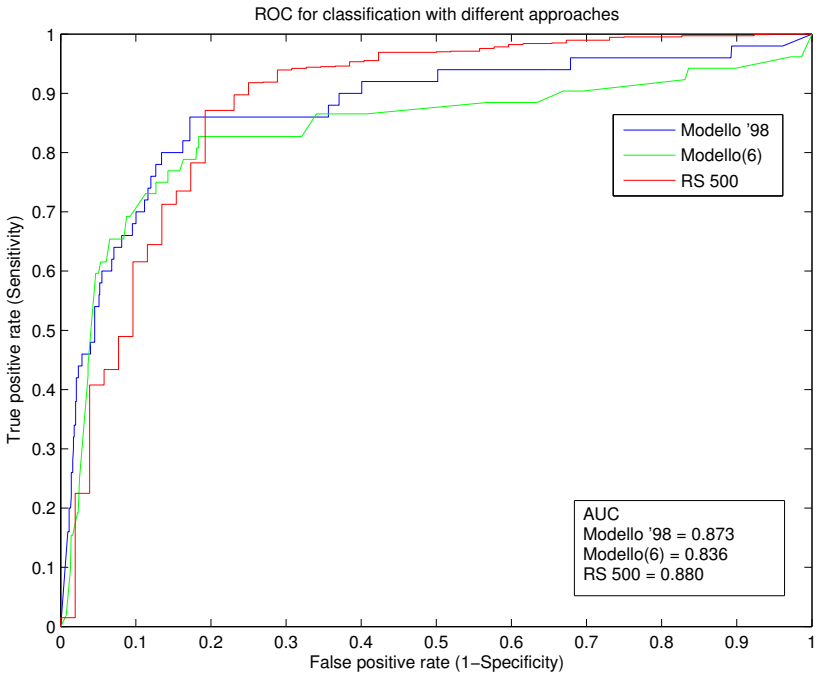


Figura 4.10: Confronto prestazioni tra i nuovi modelli ed il modello del '98

Si consideri ora la Figura 4.11 che confronta il modello '98 con gli ensemble formati dallo stesso modello logistico assieme ai nuovi classificatori creati. Anche in questo caso si cerca di fare un'analisi completa basata sui valori di AUC e sulle coppie sensibilità-

specificità. I valori di tali coppie sono riportati in Figura 4.13. L'AUC maggiore viene ottenuta con la combinazione '98+RS 500: 0.899. Con l'ensemble '98+Modello(6) l'area sotto la curva è pari a 0.892, mentre il modello del'98 ha un area inferiore, pari a 0.873. Anche in questo caso valgono le considerazioni fatte nel confronto tra Modello(6) e RS 500 con il modello '98. Fissando una specificità elevata, 95%, la sensibilità ottenuta con '98+RS 500 è di circa 40%, molto inferiore rispetto al modello '98 e '98+Modello(6) che ottengono un valore attorno al 60%. Per valori inferiori di specificità le cose cambiano, infatti fissandola al 90% si ha una sensibilità del 80% per '98+RS 500 contro il 70% realizzato con gli altri due modelli. Abbassando via via i valori di specificità gli ensemble garantiscono percentuali di sensibilità sempre migliori rispetto al modello logistico del '98. Si nota che in alcuni punti la curva di '98+RS 500 si sovrappone a quella di '98+Modello(6), ad esempio con specificità del 80% entrambi hanno una sensibilità del 90% circa, maggiore rispetto all'85% del modello '98. In maniera simmetrica a quanto detto finora a parità di elevata sensibilità, 95%, '98+RS 500 prevede una specificità del 69% molto maggiore rispetto al 58% raggiunto dall'ensemble '98+Modello(6) e al modello '98 con 50%. Con una sensibilità fissata al 90% i due ensemble hanno una specificità che si attesta attorno all'80%, ancora una volta superiore nei confronti del 72% raggiunto dal modello '98.

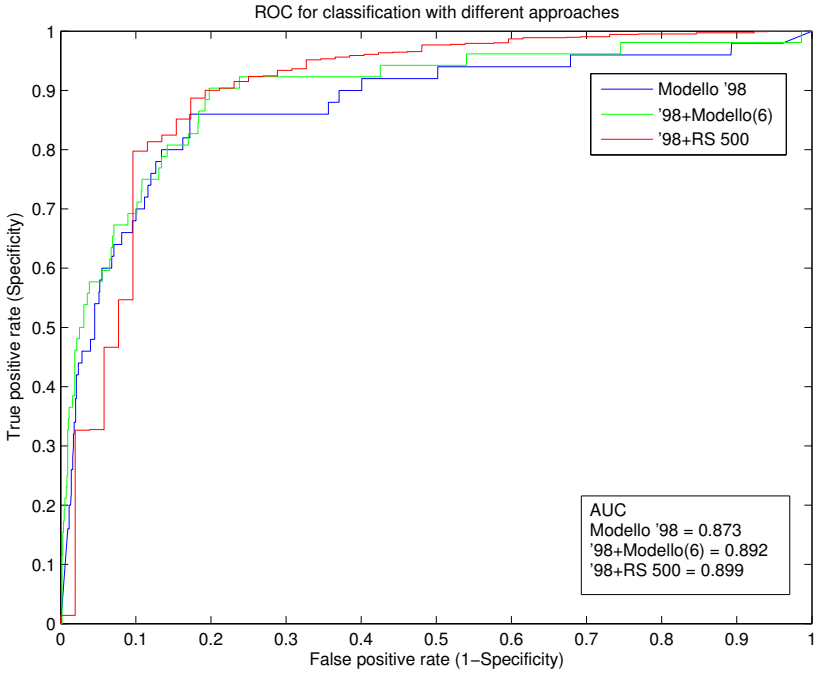


Figura 4.11: Confrontro prestazioni tra gli ensemble ed il modello '98

Infine nella Figura 4.12 sono riportate le curve di tutti i modelli analizzati finora. Questo permette a colpo d'occhio di vedere che gli ensemble funzionano meglio dei rispettivi singoli modelli che li compongono. Le loro curve ROC stanno infatti quasi costantemente al di sopra delle altre, ciò si traduce in maggiori percentuali di sensibilità e specificità. Considerevole è il miglioramento con '98+Modello(6) rispetto a Modello(6), a parità di specificità si arriva ad una differenza, in termini di sensibilità, anche del 10%. Si consideri ad esempio il valore 80% di specificità, il Modello(6) arriva a sensibilità del 80% contro il 90% raggiunto dall'ensemble. Questa differenza tra percentuali si abbassa all'aumentare del valore di specificità scelto come appare dalle Tabelle 4.13. Infatti con specificità 95% entrambi hanno una sensibilità attorno al 60%. Si nota una differenza di prestazioni anche tra '98+RS 500 ed RS 500. Per specificità inferiore al 70% le due curve sono molto simili, mentre quando il suo valore aumenta c'è una differenza ben visibile in termini di sensibilità. Ad esempio la specificità del 90% prevede una sensibilità del 80% con l'ensemble rispetto al 62% di RS 500. Tra tutti i modelli realizzati '98+RS 500 ha la ROC che più si avvicina al punto (0,1) che indica la classificazione perfetta, inoltre per valori elevati di sensibilità, ad esempio, 95% offre la maggior percentuale possibile di specificità, circa 70%, rispetto a tutti gli altri modelli considerati.

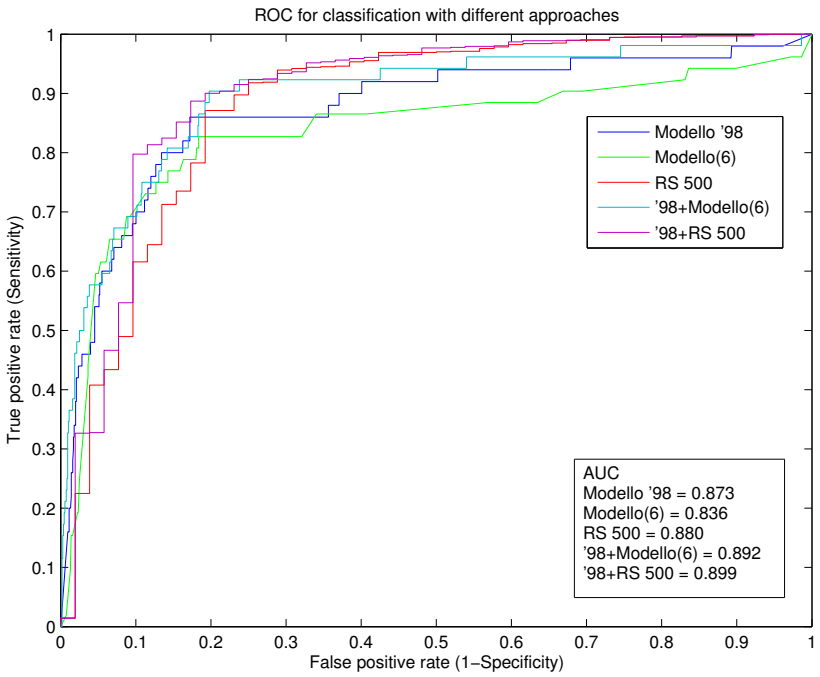


Figura 4.12: Confronto prestazioni tra tutti i modelli analizzati

Modello(6)

Specificità	Sensibilità
96.22%	39.78%
95.88%	60.45%
90.80%	70.76%
82.59%	80.76%
66.35%	86.45%
30.5%	90.38%
17.32%	94.23%

RS 500

Specificità	Sensibilità
97.0%	40.3%
91.2%	61.23%
87.10%	71.45%
81.73%	80.98%
81.52%	85.91%
73.51%	90.86%
62.64%	95.38%

Modello '98

Specificità	Sensibilità
98.0%	40.00%
95.3%	60.22%
88.21%	71.88%
87.73%	80.18%
80.17%	86.00%
72.11%	89.32%
50.42%	94.23%

'98 + Modello(6)

Specificità	Sensibilità
99.59%	40.38%
94.29%	61.56%
90.28%	71.15%
84.02%	80.77%
82.88%	84.62%
80.13%	90.38%
58.46%	94.23%

'98 + RS 500

Specificità	Sensibilità
95.34%	40.36%
89.45%	61.11%
89.11%	70.71%
89.11%	80.11%
85.33%	85.07%
81.49%	90.31%
68.79%	95.12%

Riepilogo AUC

Modello	AUC
Modello '98	0.873
Modello(5)	0.825
Modello(6)	0.836
RS 500	0.880
'98+Modello(6)	0.892
'98+RS 500	0.899

Figura 4.13: Tabelle di specificità-sensibilità riassuntive e valori di AUC

Riassumendo quanto detto non è possibile identificare un modello che primeggia in tutto rispetto agli altri. I classificatori che si comportano meglio quando si ha necessità di una specificità elevata sono Modello(6), '98+Modello(6) e modello '98. Tali classificatori, qualora vengano impostati (definita la soglia di classificazione) in modo da avere un'elevata capacità di identificare i pazienti sani, a parità di condizione, classificano meglio i record malati rispetto ai modelli basati su SVM. Con valori di specificità del 95% prevedono

una sensibilità attorno al 60%. I classificatori sviluppati con SVM sono lontani da queste percentuali, si attestano a circa il 45%. Ciò avviene esclusivamente quando si cerca di avere un modello affidabile nell'identificazione dei pazienti privi di APA. Al diminuire della specificità, perciò all'aumentare della sensibilità, le cose si invertono ed i modelli basati su SVM iniziano a funzionare meglio. Per valori elevati di sensibilità, attorno al 90% il Modello(6), il modello '98 e l'ensemble '98+Modello(6) garantiscono valori inferiori di specificità se confrontati con RS 500 e '98+500. In questo caso, quando si richiede un'elevata capacità di riconoscere i pazienti malati, a parità di condizione i modelli RS 500 e '98+RS 500 funzionano meglio in quanto riconoscono maggiormente anche i pazienti privi di APA. La differenza di specificità, a parità di sensibilità, arriva anche ad essere di 20 punti percentuali.

È da sottolineare che i modelli sviluppati con i PET (Modello(6) e '98+Modello(6)) offrono un vantaggio importante rispetto agli altri: sono facilmente comprensibili. L'albero decisionale può venire utilizzato ed interpretato in maniera molto semplice dai medici, inoltre in esso sono contenute informazioni riguardo alle variabili utilizzate per classificare i pazienti. Gli stessi valori utilizzati nei test di split possono venire valutati dai medici con un significato medico ed offrire ulteriori informazioni. Tutto ciò non accade con RS 500, esso viene considerato un modello "black box", ovvero qualcosa che riceve in input i dati del paziente e restituisce in output la classificazione senza fornire alcuna spiegazione, di facile comprensione, su come ciò avviene. Lo stesso modello del '98 non ha un elevato potere informativo riguardo alle variabili utilizzate, l'unica interpretazione può venire effettuata sui coefficienti della funzione di regressione: più elevati sono e più l'influenza di quella variabile è forte ai fini della classificazione. Perciò a parità di prestazioni (ad esempio quando si ha una specificità compresa tra 100% ed 85% la sensibilità di Modello(6) e modello '98 sono praticamente uguali) i modelli basati sui PET sono preferibili poiché offrono una quantità di informazione maggiore.

Le scelte su quale modello utilizzare vanno fatte in base alle esigenze che si hanno. Viene perciò da osservare che qualora l'obiettivo dei medici sia quello di avere un modello che garantisce un'alta la capacità di riconoscere i pazienti non affetti da APA, perciò condizioni con elevati valori di specificità, allora è conveniente affidarsi al modello del '98 oppure alla combinazione creata con i PET: '98+Modello(6). Le prestazioni di questi due classificatori si equivalgono in termini di percentuali di sensibilità e specificità, sono quelli che garantiscono la maggiore sensibilità possibile quando la specificità viene scelta molto elevata. Per quanto affermato prima è consigliabile la combinazione '98+PET perchè offre un albero decisionale con maggiori informazioni disponibili per il medico. Se invece l'obiettivo è quello di avere a disposizione un modello che fornisce la più elevata sensibilità, perciò una grande capacità d'identificare i pazienti con APA, allora la scelta ricade su RS+500 o '98+RS 500. Questi modelli permettono di avere una sensibilità molto alta, ad esempio del 95%, mantenendo la specificità a livelli ragionevoli attorno al 63% e 70%

rispettivamente. Ciò non è assolutamente possibile con gli altri tre modelli poichè per avere tali valori di sensibilità bisogna ridurre la specificità a livelli molto bassi attorno al 50%.

Da sottolineare è il fatto che il Modello(5) utilizza tutte quattro le variabili, più altre, che sono presenti in entrambi i modelli del lavoro del 1998. Il Modello(6) ne contiene invece tre. Questo dimostra la coerenza e la correttezza del lavoro svolto, inoltre conferma ulteriormente che sK, PRAbase, cAldo e basAldo sono fondamentali nell'identificazione della malattia.

Capitolo 5

Conclusioni

In questo capitolo vengono riepilogati brevemente il percorso seguito in questa tesi ed i risultati raggiunti, inoltre viene fatto un commento finale riassuntivo sul lavoro svolto. Nell'ultimo paragrafo vengono indicati eventuali sviluppi futuri che possono portare ad un miglioramento delle prestazioni raggiunte.

5.1 Sintesi dei risultati ottenuti

In questo lavoro di tesi si è cercato di sviluppare un classificatore che permetta l'identificazione dei pazienti affetti da Iperaldosteronismo Primario dovuto alla presenza dell'adenoma di Conn (APA). La precoce identificazione del sottotipo corretto di PA permette, dove possibile, di intervenire chirurgicamente in modo da guarire la patologia e correggere l'ipertensione arteriosa. Nel 1998 è stato sviluppato un modello con finalità analoghe utilizzando la Logistic Regression. Inizialmente, per lo sviluppo del nuovo lavoro, la scelta è ricaduta sui PET poiché garantiscono diversi vantaggi: un modello finale di facile interpretazione, che offre informazioni sulle variabili che consentono di identificare la malattia e che fornisce buoni risultati. I dati utilizzati per la fase di training provengono dallo studio PAPY realizzato nel 2006. Così come accade in molti problemi medici la presenza di record positivi (affetti da APA) è limitata, vi è infatti un rapporto 20:1 tra la classe di maggioranza e quella di minoranza. L'uso dei PET porta a buoni risultati ma non ottimali se confrontati con quelli raggiunti nel 1998. Per questo motivo si è provato a creare un nuovo modello utilizzando la tecnica delle SVM. Al contrario dei PET, il modello realizzato con SVM non è facilmente interpretabile dai medici, viene considerato un modello "black box": in input entrano i dati del paziente e in uscita si ottiene la classificazione senza avere informazioni su come ciò avviene. Quanto realizzato con SVM (modello RS 500) ha portato a raggiungere performance migliori rispetto a quelle ottenute con i PET (Modello(6)). Scegliendo soglie di classificazione diverse si hanno classificazioni diverse per

lo stesso record, perciò più in generale diversi valori di sensibilità e specificità. Quando si parla di scelte riguardo alla sensibilità, o alla specificità, significa che si impone una soglia di classificazione che garantisca le percentuali desiderate. Per valori elevati di sensibilità, attorno al 95%, il modello RS 500 funziona molto meglio rispetto al modello '98 e al Modello(6). In queste condizioni la specificità di RS 500 è del 60% contro il 50% del modello logistico ed il 17% del Modello(6). Al contrario, con una specificità elevata, del 90%, il modello RS 500 funziona in modo peggiore perchè la sua sensibilità si ferma al 40% contro il 60% del modello '98 e del Modello(6). Considerando che il lavoro del 1998 fornisce buoni risultati si è provato a combinarlo con i nuovi classificatori sviluppati. Gli ensemble ottenuti in questo modo funzionano meglio rispetto ai singoli modelli che li compongono. Questo vale sia in termini di AUC che di percentuali per le coppie sensibilità e specificità. Le curve ROC degli ensemble si trovano infatti quasi costantemente sopra quelle degli altri classificatori, questo si traduce in percentuali maggiori di sensibilità e specificità. In alcuni punti le curve si sovrappongono, segno che con tale soglia di classificazione le prestazioni si equivalgono.

La scelta di quale modello possa essere considerato migliore si basa però sulle necessità dei medici. Se l'esigenza è quella di avere un modello che garantisca un buon tradeoff tra sensibilità e specificità allora l'ensemble '98+RS 500 è quello preferibile. Con tale modello è possibile ottenere una coppia di 85%/85% per specificità e sensibilità rispettivamente. I restanti modelli garantiscono tutti delle coppie attorno all' 80%/85%. Questo significa che con la medesima sensibilità, la percentuale di specificità risulta essere minore. A parità di prestazioni è bene ricordare che i classificatori costruiti con i PET hanno il vantaggio di offrire un modello altamente interpretabile, cosa che non accade con il lavoro realizzato nel 1998. Qualora la necessità dei medici sia invece quella di ottenere elevati valori di sensibilità allora il modello più adatto è ancora '98+RS 500. Si arriva anche a differenze del 10, 15% tra la specificità garantita da tale modello, rispetto a quella ottenuta nel '98, quando la sensibilità è scelta molto elevata. In altri termini, quando la soglia è fissata in modo tale da garantire un'elevata capacità di riconoscere i pazienti malati, a parità di condizione, '98+RS 500 è in grado di riconoscere meglio i pazienti sani rispetto al modello del '98. Anche il Modello(6) e '98+Modello(6) sono da considerarsi migliori, sotto quest'ottica, rispetto al modello logistico. In tale confronto i migliormanti che si ottengono, in termini di valore percentuale, sono però più contenuti rispetto a quanto avviene con '98+RS 500. Con sensibilità fissata al 95% si ha una specificità del 50% per il modello '98, del 60% circa per '98+Modello(6) e RS 500. Qualora invece la necessità sia quella di ottenere un'elevata specificità, perciò un'elevata capacità di riconoscere i pazienti sani, il modello '98, il Modello(6) e '98+Modello(6) si equivalgono. Quando la specificità ha un valore del 95%, tutti e tre hanno una sensibilità attorno al 60%. I modelli basati su SVM si fermano invece, in questo caso, ad una sensibilità del solo 40%.

Da quanto detto si possono notare evidenti miglioramenti rispetto al lavoro fatto nel

1998. I nuovi classificatori realizzati sono preferibili sia quando vi è necessità di un tradeoff tra sensibilità e specificità, e sia quando si richiede un modello con un'elevata capacità di identificare i pazienti con APA. Quando invece viene richiesta elevata specificità, Modello(6) e '98+Modello(6) offrono invece prestazioni che sono in linea con il modello '98. Questa condizione sembra essere però poco interessante a livello medico visto che l'obiettivo è quello di identificare i pazienti affetti da APA e non quelli che non ne sono affetti. In ogni caso a parità di prestazioni i nuovi classifcatori basati sui PET forniscono un modello maggiormente interpretabile dai medici rispetto a quello sviluppato con la regressione logistica. In un primo incontro tenutosi con alcuni medici della Clinica Medica 4 dell'Università di Padova, tra cui il prof. Gian Paolo Rossi, è emerso il loro interesse verso un modello che permetta la più elevata sensibilità possibile. In questo modo è possibile riconoscere un'alta percentuale dei pazienti affetti da APA. Secondo questa indicazione il modello migliore è da considerare '98+RS 500. Da questo incontro è emerso inoltre che i PET realizzati offrono informazioni plausibili a livello medico. Ciò significa che i valori sui quali vengono effettuati gli split sono ragionevoli ed interpretabili anche secondo un'ottica diagnostica. Questo conferma ulteriormente la bontà dei modelli finali ottenuti: sia Modello(6) che Modello(5).

5.2 Sviluppi Futuri

Vengono ora descritti alcuni sviluppi futuri che potrebbero venire intrapresi per cercare di migliorare ed ampliare questo progetto:

- In [16] viene descritto un sistema che prevede una particolare imputazione dei missing values utilizzando EM. Il training set viene suddiviso in un numero fisso di cluster in modo da ottenere un imputazione multipla per ciascun valore mancante. Questo diverso utilizzo di EM dovrebbe fornire performance migliori rispetto al metodo di replace di missing values utilizzato in questo lavoro.

- Come detto in Sezione 4.2 è possibile provare ad implementare qualche tecnica di oversampling che permetta di costruire un training set più bilanciato da utilizzare anche con SVM. Visto che l'algoritmo SMOTE è piuttosto datato (2002), l'idea è quella di utilizzare qualche tecnica più recente, esempi possono essere [38]. Allo stesso modo può essere interessante implementare delle tecniche di undersampling più recenti e meno "radicali" del Random Subspace [39].

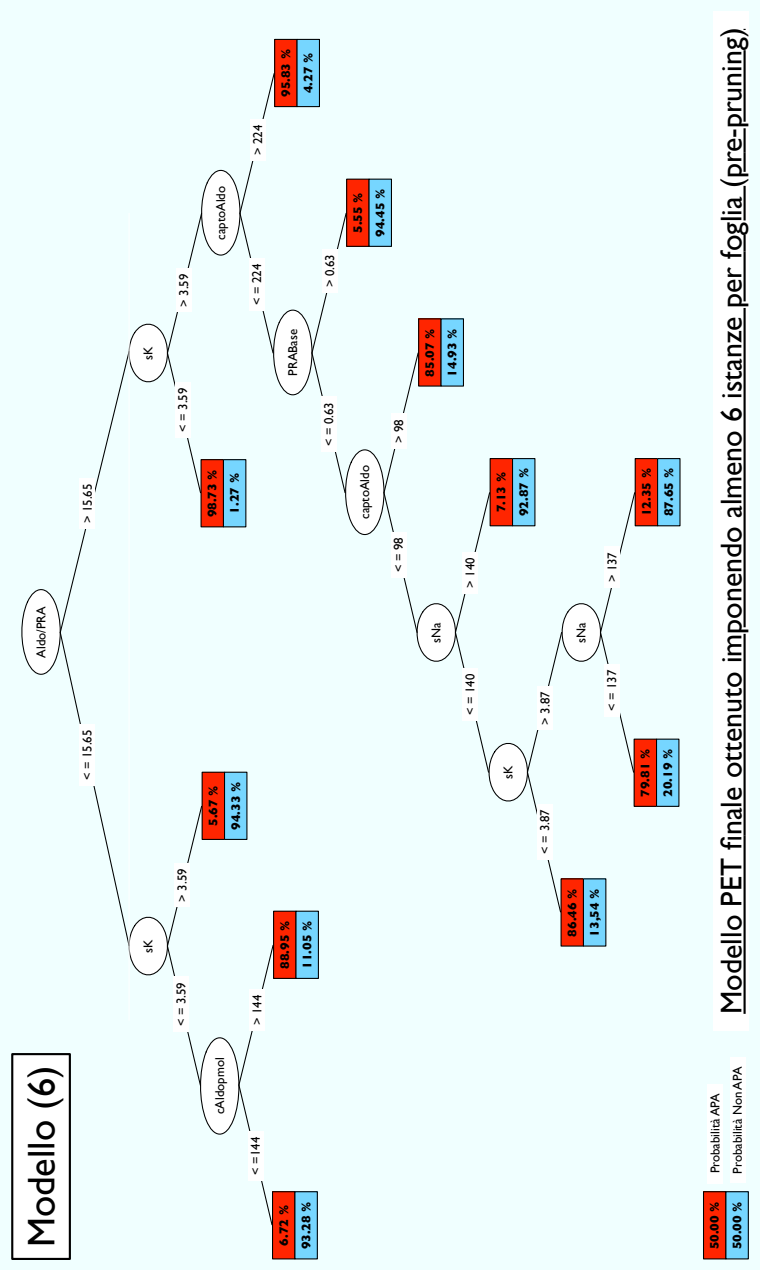
- Dato che la tecnica di ensemble learning Random Subspace ha portato ad un deciso miglioramento in termini di prestazioni, è possibile provare ad implementare altri metodi simili che prevedono la combinazione di più classificatori. RotBoost [40] è

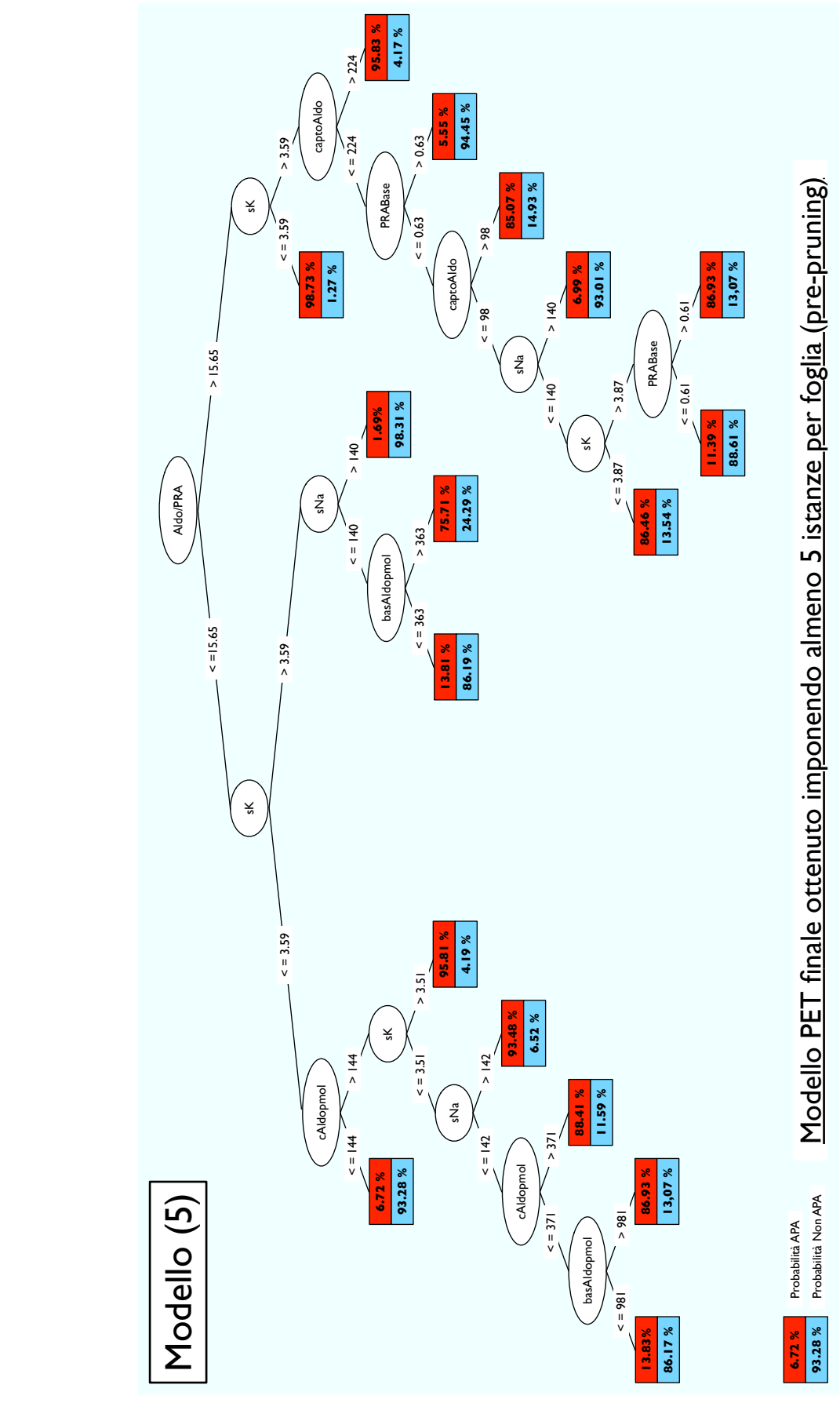
una possibile soluzione, si tratta della combinazione di due tecniche: Rotation Forest ed AdaBoost. Uno degli aspetti positivi di questo metodo è la sua implementazione open source in codice Matlab che lo rende immediatamente compatibile con il lavoro sviluppato finora.

- Qualora vengano trovati nuovi dati, dello stesso genere di quelli presenti nel PAPY, è possibile provare a ri-allenare i modelli creati per cercare di aumenterne le prestazioni. Può essere interessante utilizzare questi nuovi dati come test set per valutare il comportamento dei modelli creati su record totalmente nuovi e non usati in fase di training.

Appendice A

Modelli Finali

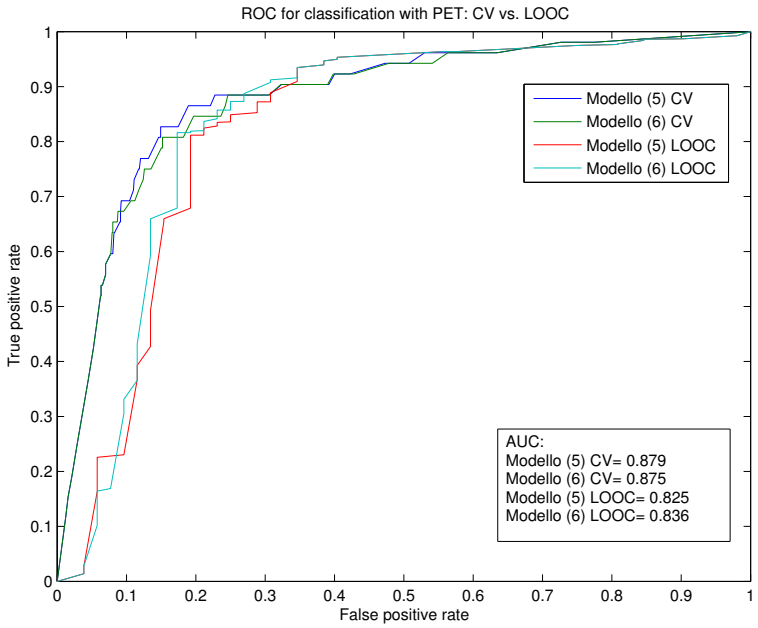



Modello (5)
Aldo/PRA
<= 15.65
> 15.65
sK
<= 3.59
> 3.59
cAldopmol
<= 144
> 144
6.72 %
93.28 %
sK
<= 3.51
> 3.51
95.81 %
4.19 %
sNa
<= 142
> 142
93.48 %
6.52 %
cAldopmol
<= 371
> 371
88.41 %
11.59 %
basAldopmol
<= 981
> 981
13.83%
86.17 %
86.93 %
13,07 %
sNa
<= 140
> 140
1.69%
98.31 %
basAldopmol
<= 363
> 363
75.71 %
24.29 %
13.81 %
86.19 %
sK
<= 3.59
> 3.59
98.73 %
1.27 %
captoAldo
<= 224
> 224
95.83 %
4.17 %
PRABase
<= 0.63
> 0.63
5.55 %
94.45 %
captoAldo
<= 98
> 98
sNa
<= 140
> 140
6.99 %
93.01 %
sK
<= 3.87
> 3.87
86.46 %
13.54 %
PRABase
<= 0.61
> 0.61
11.39 %
88.61 %
85.07 %
14.93 %
86.93 %
13,07 %
Modello PET finale ottenuto imponendo almeno 5 istanze per foglia (pre-pruning).
Probabilità APA
Probabilità Non APA
6.72 %
93.28 %

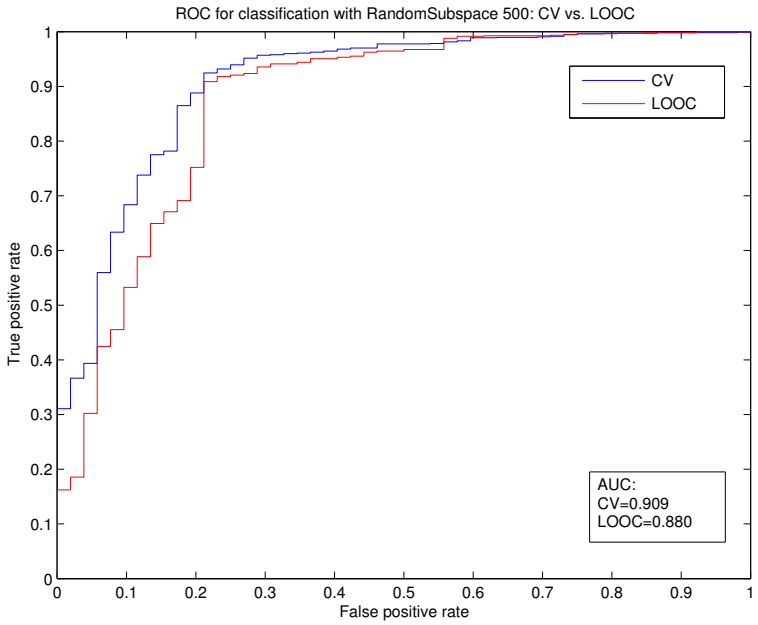
Appendice B

Cross Validation vs. Leave One Out Clinica

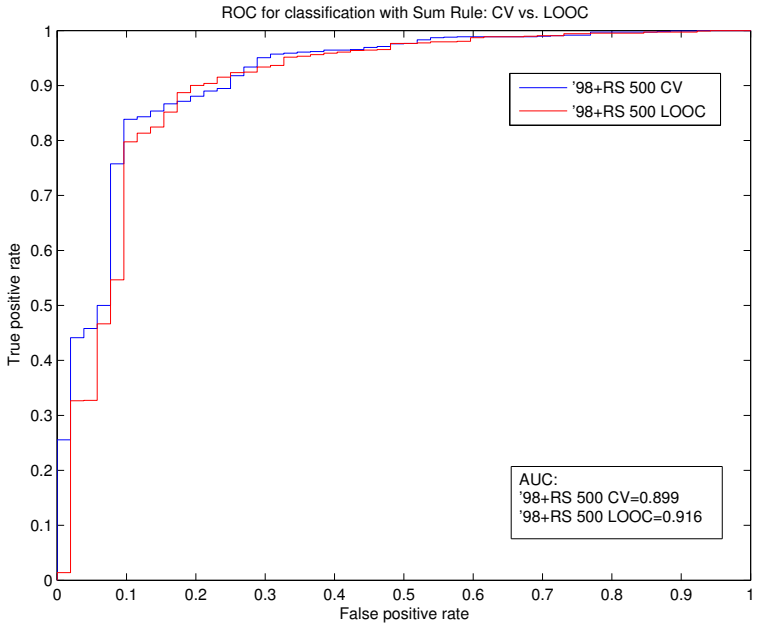
Confrontro tra le prestazioni dei modelli ottenute secondo due metodi di validazione diversi: 10-Fold Cross Validation e Leave One Out Clinica.



Confronto tra 10-Fold CV e LOOC con modelli PET



Confronto tra 10-Fold CV e LOOC con RS 500



Confronto tra 10-Fold CV e LOOC con ensemble '98+RS 500

Appendice C

Variabili disponibili

C.1 Variabili Modello '98

Variabile	Nome in PAPY
Sesso [Maschio/Femmina]	Sesso
Età [Anni]	Età
Altezza [metri]	Altezza
Peso [Kg]	Peso
* sPRA [ng/ml/h]	PRABase
cPRA [ng/ml/h]	captoPRA
* sALDO [pmol/l]	basAldo
* cALDO [pmol/l]	cAldo
Heart Rate [bpm]	Fc
SBP [mmHg]	SBP
DBP [mmHg]	DBP
* K^+ [mmol/l]	sK
Durata dell'ipertensione [Mesi]	-
BUN [mmol/l]	-
sCreatinine [μmol/l]	sCreat
Derivate	
BMI [kg/m^2]	BMI
MBP [mmHg]	MBP
$\underline{\triangle}$PRA: (cPRA-sPRA)	.
$\underline{\triangle}$ ALDO: (sALDO-cALDO)	-
sALDO/sPRA ratio	ALDO/Pra

* Utilizzate nei modelli finali

C.2 Variabili Papy

Variabile	Descrizione	Missing (%)	Min	Max	Media ± Dev.Std
Età [Anni]	Età	0	13	82	46.32± 12.01
Sesso [M/F]	Sesso	0			634/490
Peso [Kg]	Peso	2	40	148.2	77.78± 15.53
Altezza [m]	Altezza	2	1.41	2.02	1.69± 0.09
BMI [Kg/m^2]	Indice massa corporea	2	16.1	57.1	26.92± 4.58
sK [mEq/L]	Potassiemia	0	2.2	5.7	3.98± 0.44
sNa [mEq/L]	Sodiemia	1	130	152	140.68± 2.67
sCreat [μMol/L]	Cratininemia	12	35	231	82.73± 17.75
uK [mEq/24h]	Potassiura urina 24h	16	12	367	60.50± 28.15
uNa [mEq/24h]	Sodiura urina 24h	16	5	826	157.30± 82.51
SBP [mmHg]	PA sistolica seduto	1	103	220	148.13± 17.11
DBP [mmHg]	PA diastolica seduto	1	60	140	95.46± 10.04
MBP [mmHg]	PA media	2	83	167	112.92± 11.29
Fc [bpm]	Freq. Cardiaca	11	46	120	72.21± 9.98
captoSBP [mmHg]	SBP dopo Captopril	3	87.1	2.15	138.07± 17.83
captoDBP [mmHg]	DBP dopo Captopril	3	50	130	89.09± 10.30
captoMBP [mmHg]	MBP dopo Captopril	5	71	158	105.33± 11.86
captoFc [bpm]	Fc dopo Captopril	12	46	126	71.02± 9.37
PRABase [ng/ml/h]	Attività Reninica Plasmatica Basale	1	0.01	43.05	1.54± 2.27
captoPRA [ng/ml/h]	PRA dopo Captopril	1	0.01	60.08	3.21± 5.28
AldoBase [pg/ml]	Aldosterone basale	0	0.56	2260	136.91± 135.57
basAldopmol [pmol/L]	Aldosterone Basale (pmol)	0	2	6260	378.10± 375.36
captoAldo [pg/ml]	Aldosterone dopo Captopril	0	1	1194	90.37± 96.87
cAldopmol [pmol/L]	Aldosterone dopo Captopril (pmol)	0	2	5374	249.69± 267.85
cortBasale [ng/ml]	Cortisolo	5	0	520	141.85± 58.17
cortCapto [ng/ml]	Cortisolo dopo Captopril	5	0	436	117.91± 53.40
Aldo/PRA [(ng/dl)/(ng/ml/h)]	rapporto Aldo/PRA	1	0.1	3600	52.66± 213.01
TerapiaTestCapto [0;6]†	Tipo di terapia seguita	4			461/310/75/87/83/50/10
PRAB02 [ng/ml/h]	PRA basale con valori < 0.2	1	0.1	43.5	1.58± 2.37

† 0: Nessuna terapia 1: Calcio antagonista diidropiridinico 2: Calcio antagonista non diidropiridinico 3: alfabloccante (doxazosina) 4: combinazione 1+3 5: combinazione 2+3 6: Nessuna terapia

Bibliografia

[1] E. Pavan N. Rosati R. Zecchel A. Semplicini F. Perazzoli G.P Rossi, E. Rossi and A.C. Pessina. Screening for primary aldosteronism with a logistic multivariate discriminant analysis. *Clinical Endocrinology*, 49:713–723, 1998.

[2] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems*. Addison-Wesley Publishing Company, USA, 6th edition, 2010.

[3] J. Ross Quinlan. C4.5: programs for machine learning. 1993.

[4] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986.

[5] Leo Breiman, Jerom Friedman, R.A. Olshen, and Charles Stone. Classification and Regression Trees. 1984.

[6] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[7] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Burlington, MA, 3 edition, 2011.

[8] Cesar Ferri, Peter A. Flach, and Jose Hernandez-Orallo. Improving the auc of probabilistic estimation trees. pages 121–132, 2003.

[9] Foster Provost and Pedro Domingos. Tree induction for probability-based ranking. *Mach. Learn.*, 52(3):199–215, September 2003.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. pages 273–297, 1995.

[11] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, pages 61–74, 1999.

[12] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844, August 1998.

[13] Sheldon M. Ross. *Introduction to Probability Models, Ninth Edition*. Academic Press, Inc., Orlando, FL, USA, 2006.

[14] Kent A. Spackman. Signal detection theory: valuable tools for evaluating inductive learning. pages 160–163, 1989.

[15] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[16] Loris Nanni, Alessandra Lumini, and Sheryl Brahnam. A classifier ensemble approach for the missing feature problem. *Artif. Intell. Med.*, 55(1):37–50, May 2012.

[17] Amanda N. Baraldi and Craig K. Enders. An introduction to modern missing data analyses. *Journal of School Psychology*, 48(1):5–3, 2010.

[18] Arthur P. Dempster, Nan M. Laird, and Donals B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journla of the Royal Statistical Society*, 39(1):1–38, 1977.

[19] Tapio Schneider. Analysis of Incomplete Climate Data: Estimation of Mean Values and Covariance Matrices and Imputation of Missing Values. *Journal of Climate*, 14:853–871, 2001.

[20] Nathalie Japkowicz. Proceedings of the aaai'2000 workshop on learning from imbalanced data set. 2000.

[21] Frank Provost Thomas Dietterich, Dragos Margineantu and Peter Turney. Proceedings of the icml'2003 workshop on learning from imbalanced data sets. 2003.

[22] Aleksander Kolcz Nitesh V. Chawla, Nathalie Japkowicz. Special issue on learning from imbalanced datasets. *SIGKDD Explor. Newsl.*

[23] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[24] Jacek Biesiada, Wlodzislaw Duch, Adam Kachel, Krystian Maczka, and Sebastian Palucha. Feature ranking methods based on information entropy with parzen windows.

[25] Gary Weiss and Foster Provost. The effect of class distribution on classifier learning: An empirical study. 2001.

[26] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. pages 63–66, 2001.

[27] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[28] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2002.

[29] Jerome W. Conn. Primary aldosteronism, a new clinical entity. *Annals of Internal Medicine*, 44(1):1–15, 1956.

[30] Gian Paolo Rossi, Giampaolo Bernini, Chiara Caliumi, and Giovambattista Desideri et al. A prospective study of the prevalence of primary aldosteronism in 1,125 hypertensive patients. *Journal of the American College of Cardiology*, 48(11):2293 – 2300, 2006.

[31] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[32] Ayelet Eyal and Roni Bar-David. Implementing filters for imbalance dataset. `http://code.google.com/p/imbalanced/`.

[33] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

[34] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[35] Chih-Chung Chang and Chih-Jen Lin. Libsvm – a library for support vector machines libsvm. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[36] W. S. Sarle. Neural network faq. `ftp://ftp.sas.com/pub/neural/FAQ.html`.

[37] Luís A. Alexandre, Aurélio C. Campilho, and Mohamad Kamel. On combining classifiers using sum and product rules. *Pattern Recogn. Lett.*, 22(12):1283–1289, October 2001.

[38] Mikel Galar, Alberto Fernandez, Edurne Barrenechea Tartas, Humberto Bustince Sola, and Francisco Herrera. A review on ensembles for the class imbalance problem: Bagging, boosting, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(4):463–484, 2012.

[39] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory under-sampling for class-imbalance learning. pages 965–969, 2006.

[40] Chun-Xia Zhang and Jiang-She Zhang. Rotboost: A technique for combining rotation forest and adaboost. *Pattern Recogn. Lett.*, 29(10):1524–1536, July 2008.

Ringraziamenti

Voglio ringraziare tutti quelli che mi sono stati vicini in questi anni e che mi hanno permesso di concludere questo lungo percorso di studi universitari. In particolare ringrazio la mia famiglia che mi ha sempre sostenuto: mio papà, mia mamma e mio fratello. Un ringraziamento va ai miei amici che hanno sempre saputo farmi "staccare la spina" dall'ambito accademico quando necessario. Infine ringrazio i Professori: *Andrea Pietracaprina*, *Geppino Pucci*, *Carlo Fantozzi*, *Loris Nanni* e *Gian Paolo Rossi* per l'opportunità che mi hanno dato, per l'aiuto e la professionalità con la quale mi hanno seguito in questo lavoro.

Grazie,

Nicola Lazzarini