Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Sviluppo di un algoritmo di predizione dello stato di salute di un paziente con insufficienza renale

Relatori:
Prof. Valentina Alice Cauda
Prof. Eros Gian Alessandro Pasero

Candidati:
Salvatore Cavalli 241991

Marzo 2020

Sommario

Oggigiorno ll danno renale acuto ["acute kidney injury" (AKI)] è una sindrome molto frequente, soprattutto nei reparti di terapia intensiva, che si associa a elevata mortalità nonostante i progressi terapeutici. La diagnosi di AKI è spesso tardiva perché i criteri diagnostici sono basati anche sulle variazioni della creatinina sierica, il cui lento incremento ne costituisce un'importante limitazione.
Lo scopo di questo studio è lo sviluppo e la validazione di modelli di previsione a supporto alle decisioni in ambito medico, in particolare riguardanti la diagnosi precoce di AKI in pazienti ricoverati in terapia intensiva.

Nella prima parte di questo studio, dopo una consistente fase di preprocessing su un database pubblico, MIMIC-III (Medical Information Mart for Intensive Care), relativo ai parametri fisiologici di 58 mila pazienti ospedalizzati, è stato creato un dataset finale adeguato alle fasi di training e test relative ad algoritmi di apprendimento supervisionato. I pazienti presenti in quest'ultimo sono stati etichettati sulla base delle linee guida alla prevenzione, diagnosi e terapia delle sindromi di danno renale acuto ["Kidney Disease: Improving Global Outcomes" (KDIGO)].

Nella seconda parte, sulla base di quanto fatto nella prima, è stata condotta un'analisi dati mediante algoritmi di machine learning. L'algoritmo più performante sulla base di varie metriche utilizzate verrà integrato all'interno di un innovativo sistema di monitoraggio attualmente in fase di sviluppo, il quale raccoglierà misurazioni di urina ogni ora ed effettuerà dopo le prime sei la previsione sullo stato di salute renale del paziente nelle successive 12, 24 e 48 ore.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Diagnosi tardiva di insufficienza renale acuta in terapia intensiva

"È oggi evidente che le sindromi acute che coinvolgono il rene non possono più essere banalmente definite come "Insufficienza renale acuta" ma devono necessariamente fare riferimento a un complesso di condizioni in cui la possibile "insufficienza" dell'organo in questione può essere presente o coesistere con un danno strutturale, oppure può anche non essere manifesta pur alla presenza di un danno strutturale" [1]. Questo concetto internazionalmente è noto come **Acute Kidney Injury** (AKI).

L'AKI caratterizza oltre il 5% di tutte le ospedalizzazioni, circa il 20% nei pazienti in terapia intensiva e riguarda il 30% nei pazienti sottoposti a chirurgie maggiori. Si manifesta tramite una repentina perdita della funzionalità renale, un aumento di tossine nel sangue, una riduzione della emissione di urina sino all'azzeramento.

"La prevenzione dell'AKI è una delle sfide cliniche ed economiche più grandi non ancora risolte nella sanità a livello globale – ha dichiarato Claudio Ronco, MD, direttore Reparto di Nefrologia, Dialisi e Trapianti presso l'International Renal Research Institute dell'ospedale San Bortolo di Vicenza. – Una diagnosi tardiva e imprecisa può portare a danni renali irreversibili e ad una maggiore morbilità, mortalità e costi ospedalieri".

L'AKI ha un notevole peso in termini di mortalità, i cui fattori di rischio comprendono il diabete, l'età, patologie concomitanti e insufficienza renale già presente. Cause comuni sono invece complicanze chirurgiche, intossicazioni, traumi e sepsi.

Nonostante l'AKI sia reversibile, può causare un danno permanente ai reni con conseguenze convergenti in insufficienza renale terminale e dialisi. Tutto ciò porta a dei costi di gestione del paziente molto elevati che variano da un + 30% nei casi più lievi sino a un + 200% e oltre nei casi più complessi. Basti pensare che solo il costo della dialisi incide sul ricovero del paziente per circa 10.000 euro, o nel caso di terapie ancora più avanzate il costo può variare tra i 30 e i 50 mila euro [2].
La creatinina sierica (SCr) è il principale biomarcatore dell'AKI, ma sfortunatamente è un marker di lesione tardivo che ritarda la diagnosi e il trattamento [3].
Secondo i "Centers for disease control and prevention", l'AKI ha colpito circa quattro milioni di persone negli Stati Uniti nel solo 2014, ed è stata causa di centinaia di migliaia di decessi l'anno, costringendo i sopravvissuti a sottoporsi a dialisi.
In uno studio [4] è stata esaminata l'associazione tra mortalità e durata del soggiorno con stadi di lesioni renali acute e recupero renale. Complessivamente, il 22% (n = 71.486) dei pazienti ha sviluppato un danno renale acuto (Stadio I: 17,5%; Stadio II: 2,4%; Stadio III: 2%); Il 16,3% dei pazienti ha soddisfatto i criteri di lesione renale acuta entro 48 ore, con un ulteriore 5,7% dopo 48 ore di ricovero in terapia intensiva. La frequenza delle lesioni acute ai reni variava tra il 9% e il 30% nelle diagnosi di ricovero in terapia intensiva. I pazienti con danno renale acuto con elevato aumento della creatinina hanno presentato un rischio di mortalità più elevato rispetto a quelli che si sono ripresi. La diagnosi di ammissione e la gravità della malattia influenzano la frequenza e la gravità della lesione renale acuta. Le strategie per prevenire anche lievi lesioni renali acute o promuovere il recupero renale possono migliorare la sopravvivenza.

1.2 Definizione e stadiazione dell'AKI

L'AKI è definita come un'improvvisa riduzione della funzionalità renale comprendente l'insufficienza renale acuta e svariate condizioni patologiche riguardanti oltre la funzione altresì la struttura renale. L'AKI comprende un insieme di sintomi e segni clinici che possono presentarsi come condizioni non specifiche (come l'ischemia o la lesione indotta da sostanze tossiche), patologie renali specifiche (ad esempio, la nefrite interstiziale acuta, le patologie glomerulari acute e le vasculiti renali) e patologie extrarenali (come l'iperazotemia prerenale o la nefropatia ostruttiva postrenale acuta).
Ad oggi, sono state presentate e successivamente validate due definizioni, che si

basano sui valori di SCr (serum creatinine) e diuresi come viene esposto nei criteri RIFLE e AKIN. Tuttavia, per l'utilizzo in sanità pubblica, per migliorare la ricerca, per facilitare il trattamento dei pazienti e la relativa gestione si preferirebbe avere una definizione unica.

1.2.1 Criteri RIFLE

L'Acute Dialysis Quality Initiative (ADQI) group ha sviluppato a Vicenza nel 2002 un sistema di diagnosi e classificazione delle compromissioni acute della funzione renale. L'acronimo RIFLE rappresenta tre stadi di gravità crescente, Risk, Injury e Failure, e due di outcome, Loss e End-Stage Renal Disease (ESRD) [1]. La gravità dei primi tre stadi si basa sulle variazioni di SCr o di diuresi, mentre per gli ultimi due si fa riferimento alla durata della perdita della funzione renale. I criteri sono stati poi lievemente modificati senza tuttavia variazioni sostanziali, dal gruppo di lavoro AKIN di cui si farà accenno nel paragrafo 1.2.2.

Risk (rischio [di AKI]): aumento della creatinina di 1,5 volte, riduzione della velocità di filtrazione glomerulare (GFR) del 25% della diuresi a meno di 0,5 ml/kg per oltre 6 ore.
Injury (danno [renale]): raddoppio della creatinina, riduzione del GFR del 50% o della diuresi a meno di 0,5 ml/kg per oltre 12 ore.
Failure (malfunzionamento [dei reni]): aumento di tre volte della creatinina, riduzione del GFR del 75% o della diuresi a meno di 0,5 ml/kg per oltre 24 ore (oliguria) o anuria per 12 ore.
Loss (perdita [di funzione renale]): perdita completa della funzione renale che richiede terapia sostitutiva (dialisi) per più di quattro settimane.
End-stage kidney disease (insufficienza renale terminale): uremia, cioè perdita completa della funzione renale che richiede terapia sostitutiva (dialisi) per più di tre mesi.

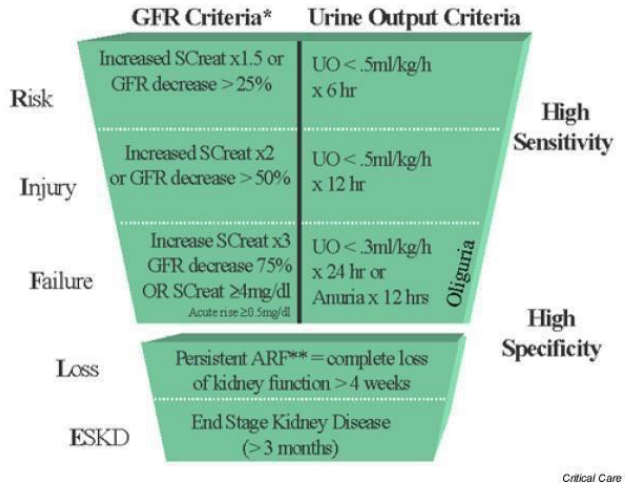


Figura 1.1. Criteri RIFLE

Più alto è il RIFLE più alto è l'indice di mortalità dei pazienti. Nel caso in cui la creatinina e la diuresi indichino due diversi livelli di gravità, lo stadio è indicato dalla funzione maggiormente compromessa.

1.2.2 Criteri Akin (Acute Kidney Injury Network)

Nel 2007 i fondatori di ADQI decisero di riunire in un summit tutte le società scientifiche interessate al problema AKI e ne scaturì un progetto per un gruppo di lavoro definito AKIN (Acute Kidney Injury Network) [5].
AKIN è una versione modificata del RIFLE, al fine di aumentare la sensibilità e la specificità della diagnosi di AKI. Per RIFLE, l'AKI dovrebbe essere sia brusco (entro 1-7 giorni) che prolungato (più di 24 ore), mentre per i criteri AKIN, l'aumento della creatinina deve avvenire entro 48 ore.
La stadiazione prevede tre livelli di crescente gravità.
Lo stadio 1 corrisponde al "Risk", ma considera anche un incremento di SCr ≥ 26.5 μmol/L (0.3 mg/dL).
Lo stadio 2 e 3 corrispondono a "Injury" e "Failure", rispettivamente.
Lo stadio 3 considera anche pazienti che richiedono la Renal Replacement Therapy (RRT) indipendentemente dallo stage (definito da SCr e/o output urinario) [6].
Gli ultimi due stadi dei criteri RIFLE non vengono presi in considerazione in quanto di eccessiva gravità [7].

Nella Figura 1.2 si riporta la combinazione dei criteri RIFLE e AKIN sopra citati:

AKIN	Diuresi (comune ad entrambi)	RIFLE	
Creatinina sierica		Classe	creatinina sierica o GFR
Fase 1 Aumento maggiore o uguale a 0.3 mg/dl (>26.5 μmol/l) o aumento maggiore o uguale da 150% al 200% (1,5 a 2 volte) rispetto al basale	Meno di 0,5 ml/kg/h per più di 6 ore	Risk	Aumento della creatinina sierica x 1.5 o diminuzione GFR >25%
Fase 2 aumento dal 200% al 300% (>2 - 3 volte) rispetto al basale	Meno di 0,5 ml/kg/h per più di 12 ore	Injury	creatinina sierica x 2 o diminuzione GFR >50%
Fase 3 Aumento di oltre il 300% (>3 volte) del basale, o maggiore o uguale a 4,0 mg/dl (≥354 μmol/l) con un aumento acuto di almeno 0,5 mg/dl (44 μmol/l) o in RRT	Meno di 0,3 ml/kg/h per 24 ore o anuria per 12 ore	Failure	creatinina sierica x 3, o >4 mg/dl (>354 μmol/l) con un aumento acuto >0,5 mg/dl (>44 μmol/l) o diminuzione GFR >75%
		Loss	insufficienza renale acuta persistente =completa perdita della funzione renale >4 settimane
		End-stage kidney disease	ESRD >3 mesi

(with the permission of Kidney International)

Figura 1.2. RIFLE vs AKIN

1.2.3 Linee guida KDIGO

La Kidney Disease Improving Global Outcomes (KDIGO), fondazione no-profit, è stata fondata nel 2003 con l'intento di "migliorare la cura e gli outcomes dei pazienti con malattia renale in tutto il mondo attraverso la promozione del coordinamento, della collaborazione, e dell'integrazione di iniziative volte a sviluppare e attuare linee guida di pratica clinica" [8].
Vista la prevalenza di AKI, la mortalità, il costo di gestione e soprattutto la mancanza di linee guida ufficiali, la fondazione KDIGO nel meeting del dicembre 2006 ha sostenuto che fosse indispensabile una cooperazione internazionale per sviluppare e attuare delle linee guida di pratica clinica nel campo dell'AKI.
Le linee guida proposte da KDIGO armonizzano i criteri di stadiazione appena citati (RIFLE e AKIN) e riconciliano diversi studi condotti da vari gruppi di lavoro accrescendo il consenso riguardante i criteri diagnostici dell'AKI.
KDIGO ha cercato di mettere in atto questi criteri provando a stilare raccomandazioni in merito ad una identificazione precoce dei possibili pazienti a rischio, accettando i criteri esistenti per la diagnosi e la stadiazione di AKI e proponendo una definizione riunificante di AKI utile per la ricerca e per le valutazioni di salute pubblica [1]:

1. AKI è definita come qualunque delle seguenti condizioni (senza grading):
   - Aumento della SCr di $\geq$0.3 mg/dl ($\geq$26.5 μmol/l) entro 48 ore.
   - Aumento della SCr a un valore $\geq$1.5 volte rispetto al valore di base che è conosciuto o è presunto essere stato almeno 7 giorni prima.
   - Volume urinario $<$ 0.5 ml/kg/ora per almeno 6 ore.
2. AKI è stadiato in termini di severità in accordo con i criteri presentati in Figura 1.3 (senza grading).
3. La causa dell'AKI deve essere determinata con precisione tempestivamente quando possibile (senza grading).

Stadio	Creatinina sierica	Diuresi
1	1,5-1,9 volte il basale oppure Incremento ≥0.3 mg/dl (≥26.5 μmol/l) rispetto al basale	<0.5 ml/kg/h per 6-12 ore
2	2,0-2,9 volte il basale	<0.5 ml/kg/h per ≥12 ore
3	3,0 volte il basale oppure incremento della creatinina sierica ≥4.0 mg/dl (≥353.6 μmol/l) oppure inizio della terapia di sostituzione renale oppure in pazienti con età <18 anni, eGFR < 35 ml/min per 1,73 m^2	0,3 ml/kg/h per ≥24 ore o Anuria per ≥12 ore

(with the permission of Kidney International)

Figura 1.3. Stadiazione AKI mediante linee guida KDIGO

eGFR rappresenta la velocità di filtrazione glomerulare stimata e può esser calcolata utilizzando l'equazione della Modification Of Diet in Renal Disease (MDRD).

$$eGFR = 175 \times (SCr)^{-1.154} \times (eta)^{-0.203} \times 0.742[se \text{ femmina}] \times 1.212[se \text{ nero}]$$

1.3 Giudizio clinico

Nonostante la diagnosi clinica di AKI sia supportata da definizioni e da un sistema di classificazione, non si dovrebbe mai escludere il giudizio clinico dato che non tutti i casi di AKI si adatteranno perfettamente alla definizione stessa.

Due problemi che potrebbero condurre alla diagnosi di AKI, tuttavia alterandone la definizione, sono definiti come: **Pseudo AKI** e **AKI atipica**.
Il primo insieme include tutti quei casi in cui sono presenti errori di dosaggio e di trascrizione di parametri, differenti risultati ottenuti a fronte di analisi in laboratori diversi e variazioni giornaliere di creatinina (fino al 10%) dipendenti anche da dieta e attività fisica. Tutti questi fattori sicuramente possono influenzare la diagnosi di AKI.
Il secondo insieme invece racchiude casi complementari alla Pseudo AKI, ovvero situazioni in cui si hanno dati poco affidabili dal momento in cui un paziente potrebbe ricevere un'eccessiva quantità di fluidi intravascolari a tal punto da abbassare in modo errato i valori di creatinina sierica o ad esempio a fronte di una trasfusione di sangue il paziente potrebbe avere un valore di SCr più vicino a quello del donatore. Anche in questo caso la diagnosi di AKI potrebbe essere influenzata da queste situazioni atipiche [7].

Capitolo 2

Stato dell'arte

Come già detto nel paragrafo 1.1 la diagnosi di AKI è spesso tardiva perchè ci si basa prevalentemente sulle variazioni di creatinina.
Da un punto di vista medico, la ricerca ha effettuato numerosi studi con l'obiettivo di individuare nuovi marcatori biochimici e compararli con quelli già noti, in modo da diagnosticare l'AKI con precocità e accuratezza, di assoluta importanza al fine di prevenirne la progressione e migliorarne la prognosi [9] [10]. Tra di essi spicca NGAL. Contemporanemeante è stato sviluppato e validato un nuovo test, Nephrocheck, a cui si deve il rilevamento di due importantissimi biomarker TIMP-2 e IGFBP-7.
Al giorno d'oggi grazie all'uso di big data e intelligenza artificiale nel campo sanitario si è assistito a un notevole incremento di vantaggi rispetto alle tradizionali analisi cliniche e le rispettive decisioni mediche. Gli algoritmi di machine learning grazie alla loro elevata precisione forniscono informazioni ultra-accurate sulla diagnostica, riuscendo a identificare con grande precisione malattie grazie alla possibilità di avere un accesso istantaneo a grosse moli di dati non alla portata di un singolo medico. La capacità di effettuare in anticipo predizioni accurate fa si che intelligenza artificiale giochi un ruolo importante anche in un tema particolarmente "caldo" nel mondo sanitario, come quello della diagnosi precoce di AKI.
L'intelligenza artificiale può aiutare quindi ad individuare i pazienti a rischio, prima ancora che inizino a mostrare i sintomi delle malattie, permettendo così ai medici di intervenire tempestivamente.
Questo capitolo ha lo scopo di illustrare dei nuovi biomarkers e conclude esponendo i vantaggi del binomio intelligenza artificiale-medicina tramite due studi di ricerca

effettuati rispettivamente da Canadian society of Nephrology e DeepMind Health.

2.1 Il ruolo di NGAL

NGAL (piccola proteina, espressa a basse concentrazioni anche nei tessuti di rene, utero, prostata, polmone, cuore, stomaco, fegato e colon [11]) che ha mostrato le migliori caratteristiche di biomarcatore diagnostico e prognostico [12][13].
Se il danno è renale, le sue concentrazioni possono aumentare nelle urine già dopo 2-4 ore dall'inizio di quest'ultimo quindi per tal motivo è considerato un precoce biomarcatore che anticipa persino la variazione della creatinina sierica, che si evidenzia quando la funzione renale è già compromessa a $\sim$50%. Nonostante NGAL presenti interessanti proprietà diagnostiche e predittive, rimangono aperte alcune problematiche analitiche che ancora limitano la sua affidabilità.
A NGAL si deve la capacità di predire la comparsa di AKI in diverse condizioni a rischio: interventi di cardiochirurgia [14], procedure contrastografiche [15][16], pazienti critici in terapia intensiva e con sepsi [17].
Se da un lato molteplici studi hanno creato entusiasmi e aspettative sulle capacità diagnostiche di NGAL, rimangono tuttavia ancora aperte alcuni problemi analitici che ne limitano la sua accuratezza diagnostica soprattutto nelle forme di danno renale subclinico [11][18][19].
Tuttavia, l'assenza di specifici valori decisionali per diversificare situazioni cliniche e di valori di riferimento ottenuti su un appropriato numero di soggetti sani, così come l'esistenza di un'eterogenea modalità di espressione dei risultati non fanno altro che ridurre l'accuratezza diagnostica di NGAL [17].

2.2 Nephrocheck

Il test NEPHROCHECK® [20] è un dispositivo diagnostico, primo nel suo genere, con lo scopo di aiutare il medico a valutare il rischio di danno renale acuto da moderato a grave nelle successive 12 ore. È stato creato per essere utilizzato su pazienti in terapia intensiva di età superiore a 21 anni.
La procedura di test consiste nell'applicazione da parte dell'operatore di un campione di urina clinica fresca o scongelata (mescolata con il coniugato etichettato

fluorescente) nella cartuccia di test, la quale verrà inserita nel misuratore ASTUTE140® per l'incubazione, la lettura, il calcolo e la visualizzazione del risultato.
Il risultato del test NEPHROCHECK®, indicante un puteggio di rischio, si basa sulle concentrazioni misurate dei due biomarcatori, TIMP-2 (inibitore tissutale delle metalloproteinasi 2) e IGFBP-7 (proteina-7 del fattore di crescita simile all'insulina).
Risultato del test NEPHROCHECK®:

$$AKIRisk = \frac{[\text{TIMP-2}] \times [\text{IGFBP-7}]}{1000} \qquad (\text{unità} = (ng/ml)^2/1000).$$

Pazienti con punteggio superiore a 0.3 sono considerati a rischio di sviluppare l'AKI.

Figura 2.1. Dispositivo Nephocheck

2.3 AKI, un aiuto dall'intelligenza artificiale

"Prediction of Acute Kidney Injury With a Machine Learning Algorithm Using Electronic Health Record Data", studio pubblicato da Canadian Journal of Kidney Health and Disease ha come obiettivo la valutazione delle prestazioni di un algoritmo di machine learning per rilevare l'insorgenza di AKI e la relativa previsione nelle 12, 24, 48 e 72 ore prima dell'inizio.
I risultati sono stati confrontati con il punteggio SOFA (Sequential Organ Failure

Assessment) [21], un sistema comunemente usato per valutare la funzione degli organi nei pazienti ospedalizzati.
In passato, il punteggio SOFA ha dimostrato di prevedere in modo indipendente il rischio e i risultati dell'AKI, e quindi costituisce un importante comparatore.
I dati utilizzati sono stati estratti dal Beth Israel Deaconess Medical Center (BIDMC; Boston, Massachusetts) e dal Stanford University Medical Center. Il primo considera solo pazienti ricoverati in terapia intensiva mentre il secondo contiene pazienti ricoverati in tutti i reparti ospedalieri.
Da entrambi i dataset sono stati selezionati pazienti maggiorenni ed oltre alle misurazioni di SCr sono state considerate anche quelle relative alla frequenza cardiaca, frequenza respiratoria, temperatura, e Glasgow Coma Scale (GCS). In questo studio è stato implementato il National Health Service (NHS) England AKI Algorithm [22], definito come gold standard, che si basa sulle linee guida KDIGO, ma vengono considerati solo i cambiamenti di SCr per determinare la presenza e la stadiazione dell'AKI.
I risultati ottenuti mediante l'algoritmo hanno mostrato un'elevata AUROC in tutti i punti temporali sperimentati. Al momento dell'insorgenza l'AUROC è di 0,872, per la previsione 12 ore prima dell'inizio è di 0,800. Infine, per le previsioni a 48 e 72 ore si sono raggiunti valori di AUROC rispettivamente pari a 0,761 e 0,728.

In un altro studio, volto a identificare i pazienti a rischio prevenendone le lesioni renali, effettuato dai ricercatori di DeepMind Health, una filiale della società di intelligenza artificiale di Google DeepMind hanno implementato un algoritmo di intelligenza artificiale [23] per anticipare la diagnosi di AKI fino a 48 ore prima, con un'accuratezza del 55,8 per cento e del 90,2 per cento nei casi abbastanza gravi da richiedere successivamente la dialisi.
L'algoritmo è stato addestrato grazie a un campione di cartelle cliniche elettroniche anonime di 703.782 adulti del Dipartimento degli affari dei veterani degli Stati Uniti (VA), contenente oltre 600.000 diversi indicatori di salute a cui il modello di intelligenza artificiale poteva attingere.
Al fine di individuare gli indicatori di salute maggiormente attendibili, è stato usato il deep learning che ha identificato 4.000 fattori rilevanti utili al calcolo del rischio molto prima che si verificasse un danno renale consentendo ai medici la diagnosi precoce di AKI due giorni prima, utilizzando quel tempo extra per prevenire o alleviare il danno, come ad esempio somministrando una maggiore quantità di liquidi

o diuretici per via endovenosa o accertandosi della non tossicità per il paziente del farmaco in uso.
Tutto ciò è stato possibile grazie alla ricchezza dei dati, afferma Dominic King, responsabile del prodotto di DeepMind Health.
Joseph Vassalotti, nefrologo e capo medico della National Kidney Foundation, ha notato che solo il 6% del campione era costituito da donne e il numero elevato di falsi positivi, suggerendo ai ricercatori di perfezionare l'algoritmo al fine di minimizzare il numero di quest'ultimi.
King ribatte sostenendo che i falsi positivi non sono una "preoccupazione reale e tangibile" principalmente perché circa la metà di essi o erano stati correttamente identificati con più di 48 ore di anticipo o "trascinavano falsi positivi", che significa " c'è stato solo un episodio di AKI e il modello continua a prevedere un aumento del rischio per un pò di tempo dopo quello." Del restante 50 percento, la maggioranza si è verificata in persone con un problema renale preesistente.

Capitolo 3

Creazione e analisi del dataset

3.1 Introduzione a MIMIC-III

MIMIC-III (Medical Information Mart for Intensive Care III) [24] è un ampio database disponibile liberamente che racchiude dati privi di identificazione relativi allo stato di salute associati a più di quarantamila pazienti ricoverati in intensive care unit (ICU) nel Beth Israel Deaconess Medical Center in un periodo compreso tra il 2001 e il 2012.
MIMIC-III estende MIMIC-II includendo i dati contenuti in MIMIC-II (raccolti tra il 2001 e il 2008) ed estendendoli con quelli appena raccolti tra il 2008 e il 2012. Oltre ad aver aggiunto nuovi dati è stata apportata una modifica nel software di gestione dei dati presso il Beth Israel Deaconess Medical Center.
Il sistema originale Philips CareVue (che ha archiviato i dati dal 2001 al 2008) è stato sostituito dal nuovo sistema di gestione dei dati Metavision (che continua ad essere utilizzato fino ad oggi).
In questo database sono incluse informazioni riguardanti dati demografici, misurazioni dei segni vitali, risultati dei test di laboratorio, procedure, farmaci, note per gli operatori sanitari, rapporti di imaging e mortalità (sia dentro che fuori dall'ospedale).
MIMIC promuove un ampio numero di studi analitici che abbraccia l'epidemiologia, il miglioramento delle regole cliniche e lo sviluppo di strumenti elettronici.
È considerevole soprattutto per tre fattori elencati qui di seguito:

- i ricercatori di tutto il mondo possono accedervi giacché disponibile liberamente

- in esso è presente un'ampia e diversificata popolazione di pazienti ricoverati in terapia intensiva
- contiene dati ad alta risoluzione temporale tra cui risultati di laboratorio, documentazione elettronica e forme d'onda del monitor al posto letto.

In Figura 3.1 si mostra la lista delle tabelle contenute in mimic-III.

Table	Children	Parents	Columns	Rows	Comments
admissions	18	1	19	58,976	Hospital admissions associated with an ICU stay.
callout		2	24	34,499	Record of when patients were ready for discharge (called out), and the actual time of their discharge (or more generally, their outcome).
caregivers	7		4	7,567	List of caregivers associated with an ICU stay.
chartevents		5	15	330,712,483	Events occuring on a patient chart.
chartevents_1			15	38,033,561	Partition of chartevents. Should not be directly queried.
chartevents_10			15	9,584,888	Partition of chartevents. Should not be directly queried.
chartevents_11			15	470,141	Partition of chartevents. Should not be directly queried.
chartevents_12			15	265,413	Partition of chartevents. Should not be directly queried.
chartevents_13			15	39,066,570	Partition of chartevents. Should not be directly queried.
chartevents_14			15	100,075,138	Partition of chartevents. Should not be directly queried.
chartevents_2			15	13,116,197	Partition of chartevents. Should not be directly queried.
chartevents_3			15	38,657,533	Partition of chartevents. Should not be directly queried.
chartevents_4			15	9,374,587	Partition of chartevents. Should not be directly queried.
chartevents_5			15	18,201,026	Partition of chartevents. Should not be directly queried.
chartevents_6			15	28,014,688	Partition of chartevents. Should not be directly queried.
chartevents_7			15	255,967	Partition of chartevents. Should not be directly queried.
chartevents_8			15	34,322,082	Partition of chartevents. Should not be directly queried.
chartevents_9			15	1,274,692	Partition of chartevents. Should not be directly queried.
cptevents		2	12	573,146	Events recorded in Current Procedural Terminology.
d_cpt			9	134	High-level dictionary of the Current Procedural Terminology.
d_icd_diagnoses	1		4	14,710	Dictionary of the International Classification of Diseases, 9th Revision (Diagnoses).
d_icd_procedures	1		4	3,898	Dictionary of the International Classification of Diseases, 9th Revision (Procedures).
d_items	8		10	12,487	Dictionary of non-laboratory-related charted items.
d_labitems	1		6	753	Dictionary of laboratory-related items.
datetimeevents		5	14	4,485,937	Events relating to a datetime.
diagnoses_icd		3	5	651,047	Diagnoses relating to a hospital admission coded using the ICD9 system.
drgcodes		2	8	125,557	Hospital stays classified using the Diagnosis-Related Group system.
icustays	8	2	12	61,532	List of ICU admissions.
inputevents_cv		4	22	17,527,935	Events relating to fluid input for patients whose data was originally stored in the CareVue database.
inputevents_mv		5	31	3,618,991	Events relating to fluid input for patients whose data was originally stored in the MetaVision database.
labevents		3	9	27,854,055	Events relating to laboratory tests.
microbiologyevents		5	16	631,726	Events relating to microbiology tests.
noteevents		3	11	2,083,180	Notes associated with hospital stays.
outputevents		5	13	4,349,218	Outputs recorded during the ICU stay.
patients	19		8	46,520	Patients associated with an admission to the ICU.
prescriptions		3	19	4,156,450	Medicines prescribed.
procedureevents_mv		5	25	258,066	Procedure start and stop times recorded for MetaVision patients.
procedures_icd		3	5	240,095	Procedures relating to a hospital admission coded using the ICD9 system.
services		2	6	73,343	Hospital services that patients were under during their hospital stay.
transfers		3	13	261,897	Location of patients during their hospital stay.
40 Tables			**534**	**728,556,685**	

Figura 3.1. Tabelle in MIMIC

Le tabelle sono collegate da identificatori che solitamente hanno il suffisso "ID". Ad esempio, HADM_ID fa riferimento a un ricovero ospedaliero univoco e SUBJECT_ID si riferisce unicamente a un paziente.
Fa eccezione ROW_ID, che è semplicemente un identificatore univoco di riga per ogni tabella. Le tabelle il cui nome inizia con "D_" indicano dizionari e forniscono definizioni per identificatori. Ad esempio, ogni riga di OUTPUTEVENTS è associata a un singolo ITEMID che rappresenta il concetto misurato, ma non contiene tuttavia il nome effettivo del farmaco. Unendo OUTPUTEVENTS e D_ITEMS su ITEMID, è possibile identificare quale concetto rappresenta un determinato ITEMID.

Di seguito si mostrano le relazioni tra le tabelle presenti, dove per una migliore visualizzazione si riporta al seguente link: `https://mit-lcp.github.io/mimic-schema-spy/relationships.html`.

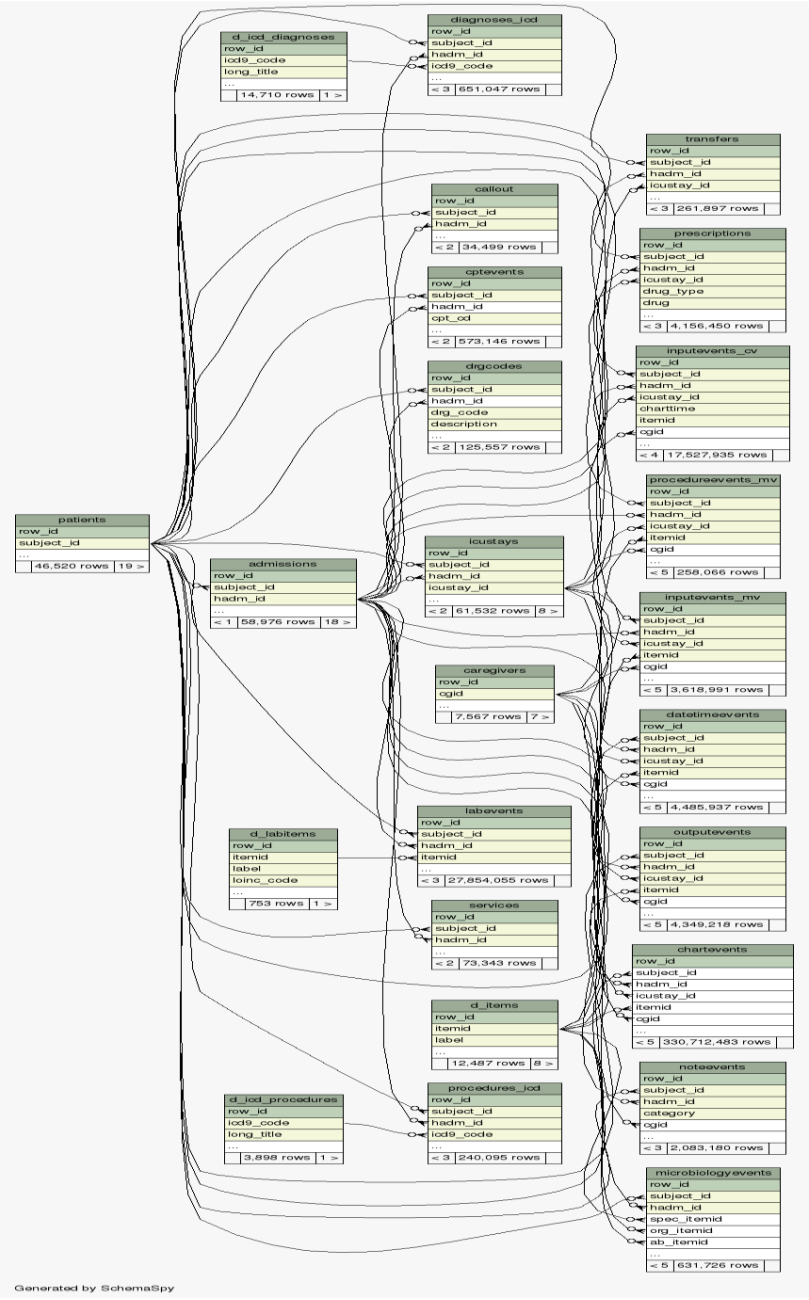


Figura 3.2. Relazioni tra le tabelle

3.2 Creazione Dataset

Partendo dal MIMIC-III e considerando solamente tabelle utili all'estrazione di misurazioni di creatinina, urina e altezza in cm ci si è posti l'obbiettivo finale di creare un unico dataset contenente per ogni record le seguenti informazioni:

- hadm_id
- icustay_id
- vettore di sei misurazioni contenente in ogni cella il rapporto fra diuresi (output urinario per ora) e peso ideale
- quattro class label binarie riferite alla presenza o meno di AKI sul momento(onset) e nelle successive 12/24/48 ore

Per quanto riguarda la classe label si è pensato di ridefinire lo stadio di AKI, includendo nel valore 0 l'assenza e la presenza di AKI allo stadio 1, e col valore 1 l'AKI allo stadio 2 e 3. Tale ridefinizione è stata attuata per tutte le class label definite nei punti temporali testati.
Per raggiungere l'obiettivo di avere un dataset avente per ogni record le informazioni descritte sopra ci si è accorti di avere problemi per calcolare l'etichetta di classe riferita all'AKI mediante i criteri KDIGO dato che essi si riferiscono a creatinina e diuresi, e le misurazioni di quest'ultime e la relativa registrazione nel MIMIC-III sono state effettuate con frequenze totalmente diverse.
Per risolvere questo inconveniente si sono condotte due analisi separate riguardanti la creatinina e la diuresi, quindi si è pensato di costruire in un primo momento due dataset, il primo riguardante per ogni paziente le misurazioni di creatinina e il suo staging di AKI relativo ai criteri KDIGO e il secondo riguardante l'output urinario e la relativa stadiazione. Successivamente si sono uniti i due dataset e si è ricalcolato lo stadio di AKI prendendo il massimo tra i due.

3.2.1 Dataset riguardante la stadiazione di AKI mediante la creatinina

Le misurazioni di creatinina sono state selezionate dalla tabella LABEVENTS tramite l'itemid 50912.

Data Output | Explain | Messages | Notifications

	row_id integer	subject_id integer	hadm_id integer	itemid integer	charttime timestamp without time zone	value character varyin	valuenum double precision	valueuom character varying	flag character varying (20)
53	1863	4	185777	50912	2191-03-16 05:42:00	0.5	0.5	mg/dL	[null]
54	1900	4	185777	50912	2191-03-17 06:00:00	0.4	0.4	mg/dL	[null]
55	1925	4	185777	50912	2191-03-18 08:00:00	0.5	0.5	mg/dL	[null]
56	1945	4	185777	50912	2191-03-19 13:00:00	0.4	0.4	mg/dL	[null]
57	1965	4	185777	50912	2191-03-20 05:00:00	0.4	0.4	mg/dL	[null]
58	1985	4	185777	50912	2191-03-21 04:30:00	0.4	0.4	mg/dL	[null]

La Figura 3.3 mostra la frequenza e altri dati statistici relativi alle misurazioni di creatinina effettuate in ospedale.

```
Media 18.85
Standard deviation 13.29
count     566681
unique      5897
top        1 day
freq       10636
Name: diff_time, dtype: object

1 day                10636
1 day 00:10:00        4218
1 day 00:05:00        4114
23:55:00              4111
23:50:00              4064
23:45:00              3940
1 day 00:15:00        3829
1 day 00:30:00        3728
23:30:00              3653
23:40:00              3209
1 day 00:20:00        3199
1 day 01:00:00        2538
1 day 00:25:00        2523
23:35:00              2475
23:00:00              2413
1 day 00:45:00        2145
23:25:00              2069
23:20:00              2052
1 day 00:40:00        2027
1 day 00:35:00        2026
23:15:00              2010
23:10:00              1768
1 day 00:50:00        1687
1 day 00:55:00        1421
1 day 01:30:00        1398
23:05:00              1398
22:30:00              1301
1 day 01:15:00        1264
1 day 01:10:00        1232
22:45:00              1228
                      ...
2 days 09:13:00          1
```

Figura 3.3. Frequenza misurazioni creatinina

L'obiettivo per entrambi i dataset è quello di avere misurazioni di creatinina e output urinario ogni ora, quindi per quanto concerne la creatinina, dal momento in

cui quest'ultima veniva misurata in media circa una volta al giorno, si è pensato di ridistribuirla mediante un algoritmo descritto qui di seguito. Per ogni record di ogni paziente:

1. Sono state riportate le date(charttime) riguardanti le misurazioni di creatinina effettuate all'ora esatta più vicina.

2. A fronte di charttime uguali se il valore di creatinina misurato è uguale, si elimina uno dei due.

3. A fronte di charttime uguali se il valore di creatinina misurato è diverso, si fa la media dei due.

4. Si creano nuovi record per compensare la mancanza di misurazioni dovuta alla natura stessa del problema.

5. I valori dei nuovi record appena creati sono frutto dell'interpolazione dei valori di creatinina tra i due estremi.

Di seguito si mostra la degenza di un paziente, con gli accorgimenti di cui sopra fino alla sua stadiazione relativa ai criteri KDIGO riguardanti esclusivamente la creatinina:

Data Output | Explain | Messages | Notifications

	row_id integer	hadm_id integer	charttime timestamp without time zone	creat double precision	admcreat double precision	admcreattime timestamp without time zone	stage_kdigo integer
704	703	100020	2142-12-02 14:00:00	0.94	0.7	2142-11-30 07:00:00	0
705	704	100020	2142-12-02 15:00:00	0.97	0.7	2142-11-30 07:00:00	0
706	705	100020	2142-12-02 16:00:00	1	0.7	2142-11-30 07:00:00	1
707	706	100020	2142-12-02 17:00:00	1.01	0.7	2142-11-30 07:00:00	1
708	707	100020	2142-12-02 18:00:00	1.03	0.7	2142-11-30 07:00:00	1
709	708	100020	2142-12-02 19:00:00	1.04	0.7	2142-11-30 07:00:00	1
710	709	100020	2142-12-02 20:00:00	1.06	0.7	2142-11-30 07:00:00	1
711	710	100020	2142-12-02 21:00:00	1.07	0.7	2142-11-30 07:00:00	1

Figura 3.4. Stadiazione AKI tramite creatinina

In Figura 3.4 si può notare che ogni record contiene le seguenti informazioni:

- *hadm_id* è l'id del ricovero ospedaliero relativo a un paziente.
- *charttime* indica l'ora in cui il valore di creatinina è misurato.

- *creat* indica il valore di creatinina stesso.
- *admcreat* indica il valore basale di creatinina, indispensabile per calcolare lo stadio di AKI.
- *admcreattime* indica l'ora in cui il valore basale di creatinina è misurato.
- *stage_kdigo* può assumere i valori discreti nell'intervallo da 0 a 3 e indica l'assenza (valore 0) o il relativo stadio di gravità di AKI.

È opportuno sottolineare che le misurazioni di creatinina contenute nel medesimo dataset contribuiranno solamente alla fase di creazione delle etichette di classe nei pazienti presenti nel dataset finale e non saranno usate come features per gli algoritmi di machine learning esposti nel capitolo 5.

3.2.2 Dataset riguardante la stadiazione di AKI mediante la diuresi

Le misurazioni riguardanti l'urina sono stati estratti dalla tabella OUTPUTEVENTS attraverso differenti "itemid" in quanto le informazioni sono state acquisite dai due sistemi informativi Carevue e Metavision.
Da un'analisi condotta riguardante le misurazioni effettuate di output urinario si evince che la frequenza massima di una misurazione è di ogni ora.

```
count      2705714
unique        1227
top       01:00:00
freq       1847563
Name: diff_time, dtype: object

01:00:00    1847563
02:00:00     459301
03:00:00      70150
00:30:00      29401
04:00:00      28758
01:30:00      22737
05:00:00      10191
02:30:00       7461
00:15:00       6611
00:45:00       6504
06:00:00       6383
01:15:00       4398
07:00:00       3510
01:45:00       2965
00:55:00       2911
00:57:00       2766
00:59:00       2745
00:58:00       2711
```

Figura 3.5. Frequenza misurazioni output urinario

L'esigenza di avere misurazioni all'ora esatta e di riempire "buchi" dovuti ad assenza di misurazioni è stata risolta con la seguente strategia:

1. Sono state riportate le date (charttime) riguardanti le misurazioni di output urinario effettuate all'ora esatta più vicina.

2. A fronte di charttime uguali, gli output urinari sono stati sommati.

3. Successivamente si effettua la sottrazione tra il charttime della misurazione corrente e il charttime della misurazione precedente e lo si memorizza nella colonna "diff_time".

4. Si calcola la diuresi (ml/h) vista come il rapporto tra urine_output e diff_time.

5. Quest'ultima viene ripartita nelle sue ore precedenti (diff_time) portando alla creazione di nuovi record.

Alla fine, si ottiene la degenza con tutte le misurazioni di urina in ml/h ad intervalli equidistanti di un'ora.
La Figura 3.6 ha lo scopo di illustrare al lettore una parte di una specifica degenza in modo da mettere in risalto il modo in cui sono state memorizzate originariamente le informazioni nel MIMIC-III:

hadm_id	charttime	urine_output
123884	2164-07-22 08:26:00	200
123884	2164-07-22 09:28:00	120
123884	2164-07-22 10:14:00	120
123884	2164-07-22 12:05:00	220
123884	2164-07-22 13:07:00	80
123884	2164-07-22 14:49:00	140
123884	2164-07-22 15:51:00	220
123884	2164-07-22 16:08:00	200
123884	2164-07-22 17:25:00	180
123884	2164-07-22 18:08:00	60

Figura 3.6. Esempio misurazioni di urina

Per soddisfare l'esigenza di avere le misurazioni all'ora esatta e colmare l'assenza di misurazioni si è applicata la strategia precedentemente descritta, il cui effetto viene mostrato in Figura 3.7:

hadm_id	charttime	urine_output	diff_time	ml/h
123884	2164-07-22 08:00:00	200	None	None
123884	2164-07-22 09:00:00	120	1	120
123884	2164-07-22 10:00:00	120	1	120
123884	2164-07-22 11:00:00	220	0	110
123884	2164-07-22 12:00:00	220	2	110
123884	2164-07-22 13:00:00	80	1	80
123884	2164-07-22 14:00:00	140	0	70
123884	2164-07-22 15:00:00	140	2	70
123884	2164-07-22 16:00:00	420	1	420
123884	2164-07-22 17:00:00	180	1	180
123884	2164-07-22 18:00:00	60	1	60

Figura 3.7. Misurazioni di diuresi ottenute

In definitiva, si mostra un esempio di degenza di un paziente e le relative misurazioni di diuresi nelle precendenti 6/12/24 ore fino alla sua stadiazione secondo le linee guida KDIGO ma facendo fede solamente ai criteri di diuresi.
In Figura 3.8 si può notare che ogni record contiene le seguenti informazioni:

- *hadm_id* è l'id del ricovero ospedaliero relativo a un paziente.
- *charttime* indica l'ora in cui il valore di diuresi è misurato.
- *urineoutput_6hr*, *urineoutput_12hr*, *urineoutput_24hr* indicano rispettivamente il totale cumulativo di diuresi nelle 6/12/24 ore precedenti.
- *ideal_body_weight_devine* indica il peso ideale del paziente calcolato secondo la formula di devine.

hadm_id integer	charttime timestamp without time zone	urineoutput_6hr double precision	urineoutput_12hr double precision	urineoutput_24hr double precision	ideal_body_weight_devine numeric	stage integer
100742	2166-03-10 15:00:00	[null]	[null]	[null]	51.2785	0
100742	2166-03-10 16:00:00	[null]	[null]	[null]	51.2785	0
100742	2166-03-10 17:00:00	[null]	[null]	[null]	51.2785	0
100742	2166-03-10 18:00:00	[null]	[null]	[null]	51.2785	0
100742	2166-03-10 19:00:00	[null]	[null]	[null]	51.2785	0
100742	2166-03-10 20:00:00	[null]	[null]	[null]	51.2785	0
100742	2166-03-10 21:00:00	690	[null]	[null]	51.2785	0
100742	2166-03-10 22:00:00	420	[null]	[null]	51.2785	0
100742	2166-03-10 23:00:00	440	[null]	[null]	51.2785	0
100742	2166-03-11 00:00:00	480	[null]	[null]	51.2785	0
100742	2166-03-11 01:00:00	480	[null]	[null]	51.2785	0
100742	2166-03-11 02:00:00	433	[null]	[null]	51.2785	0

Output Explain Messages Notifications

hadm_id integer	charttime timestamp without time zone	urineoutput_6hr double precision	urineoutput_12hr double precision	urineoutput_24hr double precision	ideal_body_weight_devine numeric	stage integer
100742	2166-03-11 03:00:00	353	1043	[null]	51.2785	0
100742	2166-03-11 04:00:00	308	728	[null]	51.2785	0
100742	2166-03-11 05:00:00	233	673	[null]	51.2785	0
100742	2166-03-11 06:00:00	188	668	[null]	51.2785	0
100742	2166-03-11 07:00:00	173	653	[null]	51.2785	0
100742	2166-03-11 08:00:00	162.5	595.5	[null]	51.2785	0
100742	2166-03-11 09:00:00	145	498	[null]	51.2785	1
100742	2166-03-11 10:00:00	125	433	[null]	51.2785	1
100742	2166-03-11 11:00:00	119	352	[null]	51.2785	1
100742	2166-03-11 12:00:00	113	301	[null]	51.2785	2
100742	2166-03-11 13:00:00	108	281	[null]	51.2785	2
100742	2166-03-11 14:00:00	95.5	258	[null]	51.2785	2
100742	2166-03-11 15:00:00	103	248	1291	51.2785	2

Figura 3.8. Esempio degenza con relativa stadiazione secondo i criteri di diuresi

- *stage* può assumere i valori discreti nell'intervallo da 0 a 3 e indica l'assenza (valore 0) o il relativo stadio di gravità di AKI.

3.3 Dataset finale

Come già accennato nel paragrafo 3.2 il dataset finale che si vorrebbe creare presenta per ogni record le seguenti informazioni: hadm_id, icustay_id, vettore di sei misurazioni contenente in ogni cella il rapporto fra diuresi (output urinario per ora) e peso ideale, quattro class label binarie riferite alla presenza o meno di AKI sul momento(onset) e nelle successive 12/24/48 ore.
Sono stati uniti i due dataset riguardanti creatinina e diuresi con il relativo livello di gravità di AKI secondo i due criteri, scegliendo come etichetta di classe il livello di gravità massimo tra i due.
Sono state considerate misurazioni di diuresi effettuate solamente in ICU e da questo

sottoinsieme sono state eliminate tutte quelle degenze che presentavano un numero di misurazioni minori di sei.
La Figura 3.9 mostra un esempio di dataset finale costruito secondo tutti gli step visti nel paragrafo 3.2.

	icustay_id	vett6	label_kdigo	label_kdigo12	label_kdigo24	label_kdigo48
1665	282685	[0.18, 0.3, 0.36, 0.24, 0.18, 0.1]	0	1	1	1
1666	282685	[0.3, 0.36, 0.24, 0.18, 0.1, 0.15]	0	1	1	1
1667	282685	[0.36, 0.24, 0.18, 0.1, 0.15, 0.1]	0	1	1	1
1668	282685	[0.24, 0.18, 0.1, 0.15, 0.1, 0.06]	0	1	1	1
1669	282685	[0.18, 0.1, 0.15, 0.1, 0.06, 0.0]	0	1	1	1
1670	282685	[0.1, 0.15, 0.1, 0.06, 0.0, 0.06]	0	1	1	1
1671	282685	[0.15, 0.1, 0.06, 0.0, 0.06, 0.06]	1	1	1	1
1672	282685	[0.1, 0.06, 0.0, 0.06, 0.06, 0.0]	1	1	1	1
1673	282685	[0.06, 0.0, 0.06, 0.06, 0.0, 0.1]	1	1	1	1
1674	282685	[0.0, 0.06, 0.06, 0.0, 0.1, 0.0]	1	1	1	1
1675	282685	[0.06, 0.06, 0.0, 0.1, 0.0, 0.0]	1	1	1	1
1676	282685	[0.06, 0.0, 0.1, 0.0, 0.0, 0.07]	1	1	1	1
1677	282685	[0.0, 0.1, 0.0, 0.0, 0.07, 0.08]	1	1	1	1
1678	282685	[0.1, 0.0, 0.0, 0.07, 0.08, 0.04]	1	1	1	1
1679	282685	[0.0, 0.0, 0.07, 0.08, 0.04, 0.0]	1	1	1	1
1680	282685	[0.0, 0.07, 0.08, 0.04, 0.0, 0.02]	1	1	1	1
1681	282685	[0.07, 0.08, 0.04, 0.0, 0.02, 0.04]	1	1	1	1
1682	282685	[0.08, 0.04, 0.0, 0.02, 0.04, 0.06]	1	1	1	1
1683	282685	[0.04, 0.0, 0.02, 0.04, 0.06, 0.06]	1	1	1	1
1684	282685	[0.0, 0.02, 0.04, 0.06, 0.06, 0.0]	1	1	1	1
1685	282685	[0.02, 0.04, 0.06, 0.06, 0.0, 0.06]	1	1	1	1
1686	282685	[0.04, 0.06, 0.06, 0.0, 0.06, 0.0]	1	1	1	1
1687	282685	[0.06, 0.06, 0.0, 0.06, 0.0, 0.05]	1	1	1	1
1688	282685	[0.06, 0.0, 0.06, 0.0, 0.05, 0.02]	1	1	1	1
1689	282685	[0.0, 0.06, 0.0, 0.05, 0.02, 0.07]	1	1	1	1
1690	282685	[0.06, 0.0, 0.05, 0.02, 0.07, 0.05]	1	1	1	1
1691	282685	[0.0, 0.05, 0.02, 0.07, 0.05, 0.0]	1	1	1	1
1692	282685	[0.05, 0.02, 0.07, 0.05, 0.0, 0.05]	1	1	1	1
1693	282685	[0.02, 0.07, 0.05, 0.0, 0.05, 0.06]	1	1	1	1
1694	282685	[0.07, 0.05, 0.0, 0.05, 0.06, 0.05]	1	1	1	1
1695	282685	[0.05, 0.0, 0.05, 0.06, 0.05, 0.05]	1	1	1	1
1696	282685	[0.0, 0.05, 0.06, 0.05, 0.05, 0.07]	1	1	1	1
1697	282685	[0.05, 0.06, 0.05, 0.05, 0.07, 0.05]	1	1	1	1
1698	282685	[0.06, 0.05, 0.05, 0.07, 0.05, 0.04]	1	1	1	1
1699	282685	[0.05, 0.05, 0.07, 0.05, 0.04, 0.04]	1	1	1	1
1700	282685	[0.05, 0.07, 0.05, 0.04, 0.04, 0.04]	1	1	1	1
1701	282685	[0.07, 0.05, 0.04, 0.04, 0.04, 0.02]	1	1	1	1

Figura 3.9. Esempio dataset finale

Il dataset finale presenta 2139599 record, 20999 hadm_id distinti e 21001 icustay_id distinti. Si deduce quindi che sono presenti solamente due degenze con due diverse ammissioni in ICU.
Ammissioni in ICU differenti riferite alla medesima degenza vengono considerate in questo studio come appartenenti a pazienti diversi.

A fronte di un conteggio sull'etichetta di classe (label_kdigo), mostrato in Figura 3.10, è emerso che il dataset risulta essere notevolmente sbilanciato per via della natura intrinseca stessa del problema, giacchè essere ricoverati in terapia intesiva non implica essere affetti da AKI.

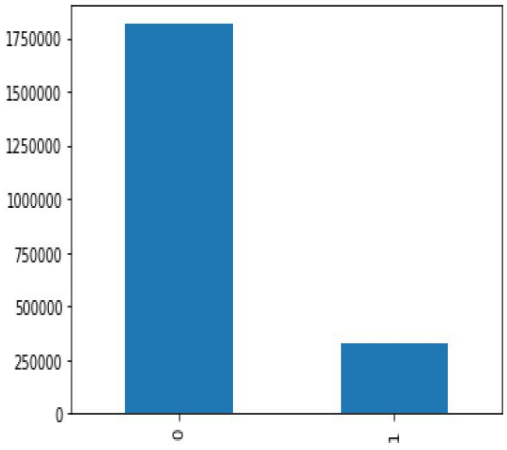


Figura 3.10. Conteggio etichetta di classe

3.4 Analisi degenze

Di seguito si riporta una breve analisi riguardante le degenze prese in carico dall'inizio di questo studio raffinate passo dopo passo in base a criteri di età, durata minima ospedaliera, intervallo temporale riguardante le misurazioni adiacenti e altezza dei pazienti indispensabile per il calcolo del peso ideale.

- Degenze totali: 58976
- Degenze relative a pazienti maggiorenni: 47464
- Degenze relative a pazienti maggiorenni di durata maggiore o uguale a 6 ore: 47270
  Le degenze relative a pazienti maggiorenni di durata maggiore o uguale a 6 ore e aventi un intervallo temporale tra due misurazioni adiacenti maggiore di un giorno sono in totale 3190. Si è notata la presenza di intervalli tra le misurazioni anche di 139 giorni (ad esempio il paziente con hadm_id 115396).

Ovviamente questi tempi così lunghi identificano un'assenza di monitoraggio di urina, quindi tali degenze sono state eliminate.

- Degenze relative a pazienti maggiorenni di durata maggiore o uguale a 6 ore e assenza massima di misurazioni pari a un giorno: 44.080

La stadiazione tramite i criteri di diuresi presuppone obbligatoriamente la presenza di una misurazione di peso in ogni paziente.
Tuttavia, dato che la maggioranza dei pazienti presentava diverse misurazioni di peso che variavano drasticamente a distanza di poche settimane, ci si è rivolti a un noto Nefrologo Dirigente e supervisore della U.O di Nefrologia dell'ospedale San Bortolo di Vicenza, autore anche di aver tradotto le linee guida KDIGO, e sotto suo consiglio si è scelto di calcolare il peso ideale di ogni paziente anziché considerare la media delle misurazioni dei pesi presenti o il primo peso registrato.
Si è usata la formula di Devine per il calcolo di quest'ultimo.
Per gli uomini:

$$IdealBodyWeight(inkg) = 50 + 0.91 \times (height[cm] - 152.4)$$

Per le donne:

$$IdealBodyWeight(inkg) = 45.5 + 0.91 \times (height[cm] - 152.4)$$

Si sono considerate solamente persone in cui fosse presente almeno una misurazione di altezza e a fronte di più misurazioni di altezza si è scelta la prima. Si sono considerate persone con altezza compresa tra 1.30 metri e 2 metri.

- Degenze relative a pazienti maggiorenni di durata maggiore o uguale a 6 ore, con assenza massima di misurazioni pari a un giorno e altezza compresa tra 1.30 metri e 2 metri: 24081
  Da quest'ultime si sono considerate solamente quelle con ammissione nel reparto di terapia intensiva (ICU).

- Degenze in ICU utilizzate per la fase di machine learning: 21001

La Figura 3.11 mostra visivamente i filtri appena descritti e applicati progressivamente a partire dal MIMIC-III fino alla creazione del dataset finale.

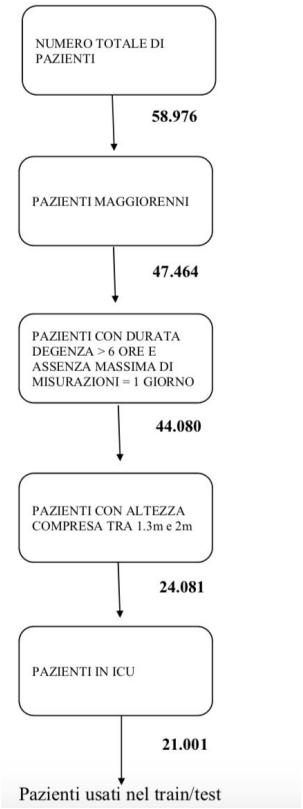


Figura 3.11. Analisi pazienti

Su un'analisi demografica effettuata sui pazienti prensenti nel dataset si evince che in termini di sesso presenta il 59,9% rappresentato dagli uomini ed il rimanente 40,1% dalle donne, quindi la popolazione risulta essere abbastanza bilanciata.
In termini di età, ovviamente nei reparti di terapia intensiva la stragrande maggioranza della popolazione è caratterizzata da pazienti anziani.

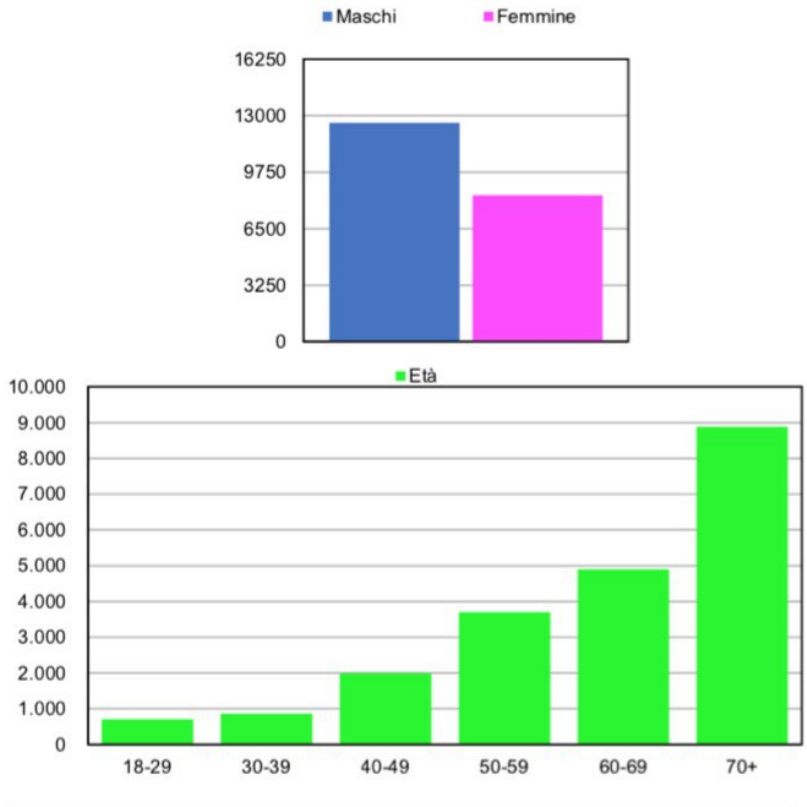


Figura 3.12. Analisi demografica dei pazienti

Caratteristiche	**BIDMC (%)**
Genere	
Uomo	59,90
Donna	40,10
Età (anni)	
18-29	3,33
30-39	4,07
40-49	9,46
50-59	17,59
60-69	23,26
70+	42,29

Tabella 3.1. Analisi demografica dei pazienti

Capitolo 4

Metriche di prestazione

In questo capitolo verrà presentata una descrizione di ogni metrica utilizzata per valutare le perfomance dei modelli di classificazione, visti come test diagnostici, che saranno trattati in dettaglio nel capitolo 5.
I test diagnostici ideali riescono a discriminare in modo ottimale due popolazioni (malati e sani) mutuamente esclusive ovvero non sovrapponibili.
Quello che realmente succede però è che le due popolazioni tendono a sovrapporsi anche in parte, e ciò porterà il test a identificare come positivi alcuni pazienti sani (Falsi Positivi) e come negativi alcuni pazienti invece malati (Falsi Negativi).
Quindi per uno specifico test diagnostico che non riesce a discriminare nettamente i malati dai sani, è opportuno che venga calcolato il grado di incertezza della classificazione.
Se il test diagnostico in questione da come risultato una variabile binaria rappresentante la condizione malato/sano, è sufficiente che vengano calcolate le seguenti metriche:

- sensibilità
- specificità
- potere predittivo positivo
- potere predittivo negativo
- accuratezza

Per buona prassi è indispensabile raccogliere i risultati ottenuti dopo una classificazione nella cosiddetta «matrice di confusione» che permette di confrontare il

valore predetto con quello reale. La matrice di confusione notifica il numero dei Veri Positivi, Veri Negativi, Falsi Positivi e Falsi Negativi, e viene utilizzata per prevedere il risultato di un classificatore binario che produce un output con due valori di classe chiamati anche etichette, come malato/sano per input noti.

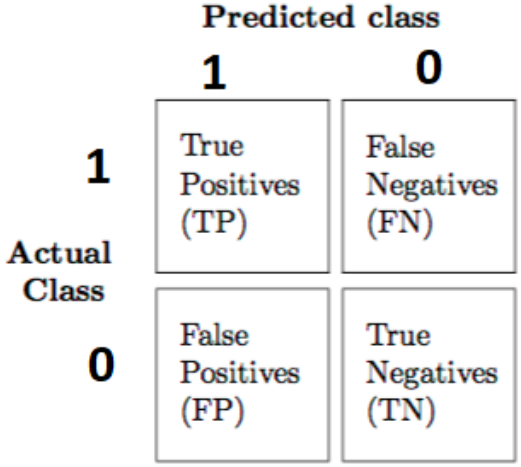


Figura 4.1. Matrice di confusione

La classe di interesse viene di solito indicata come "positiva" mentre l'altra come "negativa". Con veri positivi si intendono quei pazienti che risultano positivi al test e in cui è presente veramente la malattia. I veri negativi invece sono quei pazienti in cui il test da esito negativo e sono veramente non malati. Si può facilmente dedurre allora che i falsi positivi e i falsi negativi sono i pazienti che risultano rispettivamente positivi/negativi al test ma che realmente sono sani/malati.
Da qui in poi per indicare Veri Positivi, Veri Negativi, Falsi Positivi e Falsi Negativi ci si servirà rispettivamente delle sigle TP (true positive), TN (true negative), FP (false positive) e FN (false negative).

4.1 Sensitivity e specificity

La sensibilità e la specificità sono misure statistiche calcolate dopo l'esecuzione di un test di classificazione binaria, ampiamente utilizzate in medicina.

La sensibilità (chiamata anche tasso di positività reale o richiamo) indica la proporzione dei malati che sono correttamente identificati come tali.

$$Sensitivity = TP/(TP + FN)$$

La specificità (detta anche tasso negativo reale) misura la percentuale dei pazienti sani che risultano negativi al test.

$$Specificity = TN/(TN + FP)$$

Test medici con valori di sensibilità e specificità superiori al 90% vengono considerati con un'alta credibilità. La sensibilità e la specificità sono misure indipendenti dalla prevalenza della malattia.
A differenza di questi ultimi, i poteri predittivi sono strettamente dipendenti dalla frequenza della malattia di interesse.

Il potere predittivo positivo, definito come $PPV = TP/(TP + FP)$, indica la proporzione di pazienti malati tra tutti quelli che sono positivi al test. Ci si riferisce a quest'ultimo anche con il termine precision.

Il potere predittivo negativo, $NPV = TN/(TN + FN)$, indica la proporzione di pazienti sani tra tutti quelli che sono negativi al test.

4.2 Accuracy

L'accuratezza è una misura espressa come il rapporto dei soggetti correttamente classificati (TP + TN) tra tutta la popolazione (TP + TN + FP + FN).

$$Accuracy = (TN + TP)/(TN + TP + FN + FP)$$

L'accuratezza diagnostica è influenzata dalla prevalenza della malattia, infatti con la stessa sensibilità e specificità l'accuratezza di un determinato test aumenta man mano che diminuisce la prevalenza della malattia. Tuttavia, ciò non significa che il test sia migliore se viene applicato a una popolazione con bassa prevalenza della malattia.

4.3 ROC

Quando un modello fornisce come output un valore di probabilità, è possibile calcolarne la Receiver Operating Characteristics (ROC), grazie alla quale è possibile mettere a confronto e quindi selezionare i classificatori in base alle loro prestazioni. Questa tecnica è utilizzata già da molto tempo in campo medico, e di recente è stata presa in considerazione nel Machine learning.
La ROC viene usata solitamente per classificazioni binarie, non escludendo tuttavia il suo utilizzo in problemi multi-classe.
Una curva ROC è un grafico bidimensionale in cui le coordinate di ogni punto hanno un'ascissa rappresentata dal tasso di falsi positivi (pari a 1-specificity) e un'ordinata rappresentante la sensitivity (indicante il tasso di veri positivi).

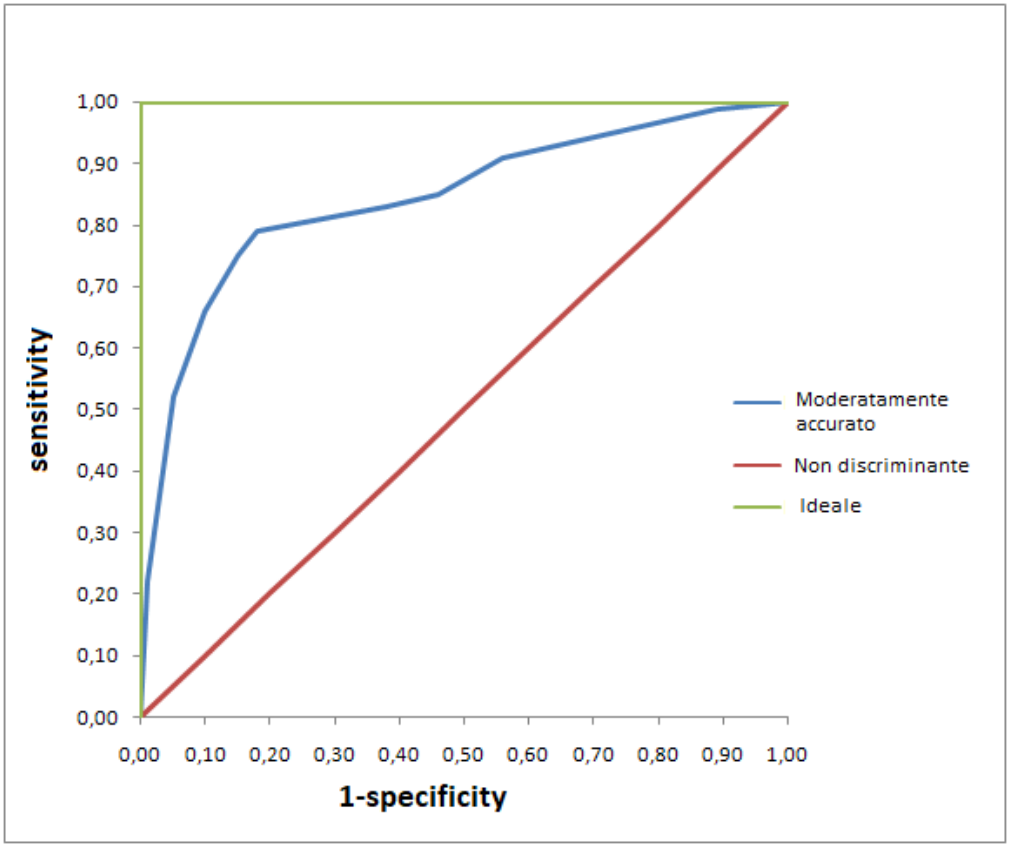


Figura 4.2. Curva ROC

Più la curva è vicina all'angolo in alto a sinistra, maggiore è l'area sotto la curva, migliore è il test nel discriminare tra malato e non malato.

In questo studio per ogni modello si è calcolata la curva ROC. Per permettere facilmente di confrontare più classificatori, la ROC viene sintetizzata in un singolo valore scalare rappresentante le prestazioni di questi ultimi.
Si parla quindi di Area Under the roc Curve (AUC), indicante la porzione di un'area di un quadrato di lato unitario il cui valore è compreso tra 0 ed 1.
Per l'interpretazione dei valori dell'area sottostante la curva ROC è possibile riferirsi alla classificazione proposta da Swets [25]:

1. AUC=0.5 il test non è informativo;
2. 0.5<AUC$\leq$ 0.7 il test è poco accurato;
3. 0.7<AUC$\leq$0.9 il test è moderatamente accurato;
4. 0.9<AUC<1.0 il test è altamente accurato;
5. AUC=1 test perfetto.

La ROC gioca un ruolo cruciale in fase di ottimizzazione in quanto consente di individuare il cosiddetto best cut-off (valore soglia ottimale), ovvero quel valore del test che riesca a massimizzare la differenza tra i veri positivi e i falsi negativi.
Come già detto precedentemente non esiste un test che riesce a distinguere nettamente i pazienti sani dai malati. È ovvio che se la soglia per definire un soggetto come positivo o negativo viene innalzata o ridotta si introdurrà un rischio maggiore o minore di falsi positivi e negativi. Tutto ciò ha un impatto notevole su sensitivity e specificity del test.
Ad esempio, un alto valore di cut off aiuta a identificare correttamente la maggior parte dei non malati, caratterizzando il test da un'elevata specificity (si avranno quindi pochi falsi positivi), ma contemporaneamente conferirà al test una bassa sensitivity che si traduce in un aumento di falsi negativi.
Contrariamente un basso valore di cut off permetterà di identificare in maniera corretta la maggioranza della popolazione caratterizzata dai pazienti affetti dalla malattia, dando al test una sensibilità elevata (ovvero saranno presenti pochi falsi negativi) e allo stesso tempo sarà colpevole di sottostimare la proporzione dei pazienti sani, caratterizzando il test con una bassa specificità (numero elevato di falsi positivi).
L'obiettivo del test caratterizza la decisione di scegliere un cut off in maniera tale da preferire un'alta specificità o un'alta sensibilità.

Supponendo di voler identificare una situazione frequente e curabile si preferirà avere un'alta sensibilità a discapito di una minore specificità, ovvero si cercherà di non perdere nessun caso anche a costo di avere un maggior numero di falsi positivi.

4.4 Precision Recall Curve (PRC)

Un'alternativa a una curva ROC è una precision recall curve (PRC). È usata meno frequentemente delle curve ROC, ma la PRC potrebbe essere una scelta migliore per dataset sbilanciati.
In questo studio infatti si è deciso di tracciare la PRC come supporto alla curva ROC, in quanto il dataset finale risulta essere sbilanciato dal momento che tutti i pazienti ospedalizzati non sono soggetti all'insorgenza di AKI.
Una curva di richiamo di precisione mostra la relazione tra precisione (valore predittivo positivo) e richiamo (o sensibilità) per ogni possibile cut-off.

La PRC è un grafico come quello mostrato in Figura 4.3 con:
- L'asse x che mostra il richiamo (sensibilità = TP / (TP + FN))
- L'asse y che mostra la precisione (valore predittivo positivo = TP / (TP + FP))

Più la curva è vicina all'angolo in alto a destra, migliore è il test nel discriminare tra malato e non malato.

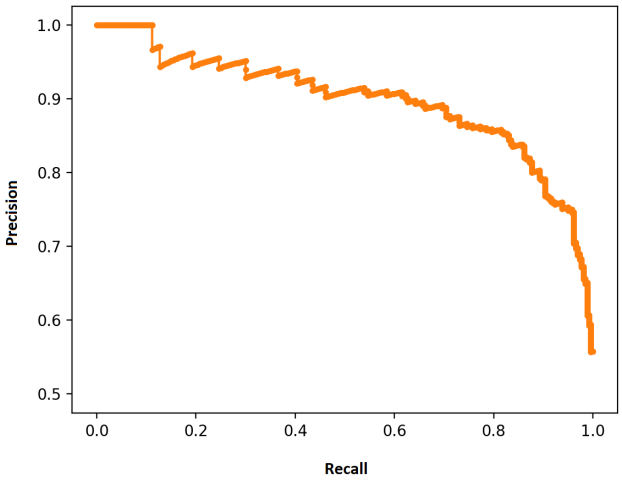


Figura 4.3. PRC

La principale differenza tra le curve ROC e le curve di richiamo di precisione è che il numero di risultati veri negativi (TN) non viene utilizzato per creare la PRC. Riguarda solo la previsione corretta della classe di minoranza, classe 1.

Curva	Asse X		Asse Y	
PRC	sensibilità	TP / (TP + FN)	precisione	TP / (TP + FP)
ROC	1-specificità	FP / (FP + TN)	sensibilità	TP / (TP + FN)

Tabella 4.1. PRC vs ROC

4.5 Diagnostic Odds Ratio (DOR)

Nei test medici con classificazione binaria il diagnostic odds ratio, noto con l'acronimo DOR, è un'altra metrica che dipende in modo significativo dalla sensibilità e dalla specificità di un test.
Il DOR di un test è il rapporto delle probabilità di positività nella malattia rispetto alle probabilità di positività nello stato "sano".
Quindi valgono le seguenti formule:

$$DOR = \frac{TP/FN}{FP/TN} = \frac{sens/(1-sens)}{(1-spec)/spec} \tag{4.1}$$

Alternativamente, il DOR può essere interpretato rapporto tre le probabilità di malattia nei pazienti in cui il test da esito positivo rispetto alle probabilità di malattia nei negativi del test.

$$DOR = \frac{TP/FP}{FN/TN} = \frac{PPV/(1-PPV)}{(1-NPV)/NPV} \tag{4.2}$$

Vi è quindi una stretta relazione tra DOR e rapporti di verosimiglianza:

$$DOR = \frac{TP/FP}{FN/TN} = \frac{LR(+)}{LR(-)} \tag{4.3}$$

Il valore di un DOR può variare da 0 a infinito, in cui i valori più alti stanno a indicare migliori performance del test discriminatorio.

Se il valore di un DOR è pari a 1, è implicito che il test non riesce a discriminare i pazienti con presenza di disturbo da quelli senza. Per valori invece inferiori a 1, il test ha un'interpretazione errata.
Si può concludere dicendo che il DOR non è dipendente dalla prevalenza della malattia, e che due test che hanno un DOR identico possono presentare sensibilità e specificità molto diverse, con diverse conseguenze cliniche.

4.6 Likelihood ratio (LR)

Nei paragrafi precedenti si è detto che ad ogni specifico test corrispondono sensitivity e specificity. Grazie a queste due metriche, e conoscendo la prevalenza della malattia su una determinata popolazione, è possibile risalire al numero di veri e falsi positivi, veri e falsi negativi e da questi calcolare il valore predittivo positivo e negativo del test in questione. Si è anche visto che mettendo in relazione sensitivity e specificity in un grafico si crea la cosiddetta curva ROC.

La likelihood ratio (LR), chiamato anche rapporto di verosimiglianza è un modo alternativo di mettere in relazione sensibilità e specificità tra loro. LR è una misura molto utile perché serve ad analizzare la validità di un test. Vi sono due tipologie di LR, una positiva (LR+) e una negativa (LR−).

LR+ è un rapporto probabilistico, in cui rispettivamente il numeratore indica la probabilità condizionata che il test sia positivo dato che il soggetto è affetto da malattia e il denominatore la probabilità condizionata che il test sia positivo dato che nel paziente è assente la malattia.
In altri termini LR+ fornisce un'indicazione su quanto è più probabile che il risultato del test fornisca un esito positivo in soggetti in cui la malattia è presente, rispetto a quelli senza.
LR+ è calcolato secondo la seguente formula:

$$LR+ = \frac{sensibilità}{(1 - specificità)} = \frac{Pr(T+|D+)}{Pr(T+|D-)}$$

$T+$ o $T-$ indica che il risultato del test è rispettivamente positivo o negativo , mentre $D+$ o $D-$ indicano che la malattia è presente o assente.
LR+ solitamente fornisce un valore superiore a 1 in quanto è più probabile che

l'esito positivo del test si verifichi in pazienti malati rispetto a quelli sani. Buoni test diagnostici presentano un valore di LR+ maggiore di 10. Maggiore è il valore di LR+, più il test è decisivo nell'indicare una malattia e ciò gli permette quindi di configurarsi come miglior indicatore per una diagnosi dominante.

Specularmente, il rapporto di verosimiglianza per i risultati negativi del test (LR−) è definito come il rapporto tra la probabilità condizionata che il test sia negativo dato che il paziente sia malato e la probabilità condizionata che il test abbia esito negativo in presenza di un paziente sano.
LR− è calcolato secondo la seguente formula:

$$LR- = \frac{(1 - sensibilità)}{specificità} = \frac{Pr(T-|D+)}{Pr(T-|D-)}$$

Contrariamente a LR+ che è determinate per l'indicazione di una malattia, LR− è un buon indicatore nell'escludere che la malattia sia presente.
Solitamente LR− presenta valori inferiori a 1, dal momento che è meno probabile che in soggetti malati il test abbia esito negativo rispetto a soggetti sani. Minore è il valore di LR−, maggiore sarà allora il contributo del test nell'escludere che nel soggetto sia presente la malattia.

Le metriche precedentemente descritte sono esposte nella tabella riassuntiva 4.2.

Metriche	**Formula**
Sensitivity	TP / (TP + FN)
Specificity	TN / (TN + FP)
Potere predittivo positivo	TP / (TP + FP)
Potere predittivo negativo	TN / (TN + FN)
Accuracy	(TN + TP) / (TN+TP+FN+FP)
DOR	(TP/FP) / (FN/TN)
LR+	sensitivity / (1 - specificity)
LR−	(1 - sensitivity) / specificity

Tabella 4.2. Riepilogo delle metriche

Capitolo 5

Analisi dei dati mediante tecniche di classificazione

Analisi dei dati mediante tecniche di classificazione Lo scopo di questo capitolo è applicare tecniche di apprendimento supervisionato al dataset costruito e descritto in dettaglio nel Capitolo 3.
L'apprendimento automatico (machine learning) è una branca dell'informatica nata a partire dall'intelligenza artificiale. Il suo punto di forza rispetto ad altri tipi di analisi sta nella capacità di scovare pattern nascosti e predire i risultati di input futuri.
Gli algoritmi di machine learning, rispetto agli algoritmi iterativi in cui le operazioni sono dichiarate esplicitamente, prendendo in prestito concetti teorici probabilistici riescono a valutare e migliorare i modelli statistici.
Esistono due definizioni di Machine Learning.
Arthur Samuel l'ha descritto come: "Il campo di studio che dà ai computer la possibilità di apprendere senza essere programmato esplicitamente". Questa è una definizione più vecchia e informale.
Tom Mitchell ha fornito una definizione più moderna [26] : "Si dice che un programma per computer apprende dall'esperienza E in relazione ad alcune classi di compiti T e alla misura della prestazione P, se le sue prestazioni nei compiti in T, misurate da P, migliorano con l'esperienza E".
Un qualsiasi problema di machine learning può appartenere a una delle seguenti categorie:

1. Apprendimento supervisionato

2. Apprendimento senza supervisione

Per apprendimento con supervisione si intende un algoritmo di apprendimento automatico che utilizza un set di dati di addestramento in cui sono incluse le etichette di classe per effettuare previsioni.
In altre parole, nel campo dell'apprendimento con supervisione la logica di classificazione viene data alla macchina tramite input, al contrario nell'apprendimento non supervisionato è demandato alla macchina il task di trovarne una.
L'apprendimento supervisionato può essere spiegato con un semplice esempio: se un medico ha bisogno di sapere se in un paziente è presente una determinata malattia, all'algoritmo verranno presentati una serie di esempi precedenti, dove l'esito sano/malato è già stato definito. Sulla base di tali esempi, l'algoritmo si allena a riconoscere quindi la presenza o meno della malattia in soggetti non ancora definiti. Al contrario, nel caso di apprendimento non supervisionato, all'algoritmo non viene consegnata la logica di classificazione.
L'apprendimento supervisionato è diviso in due categorie:

1. Regressione: in cui si cerca di predire i risultati all'interno di un output continuo.

2. Classificazione: in cui si cerca di predire i risultati in un output discreto.

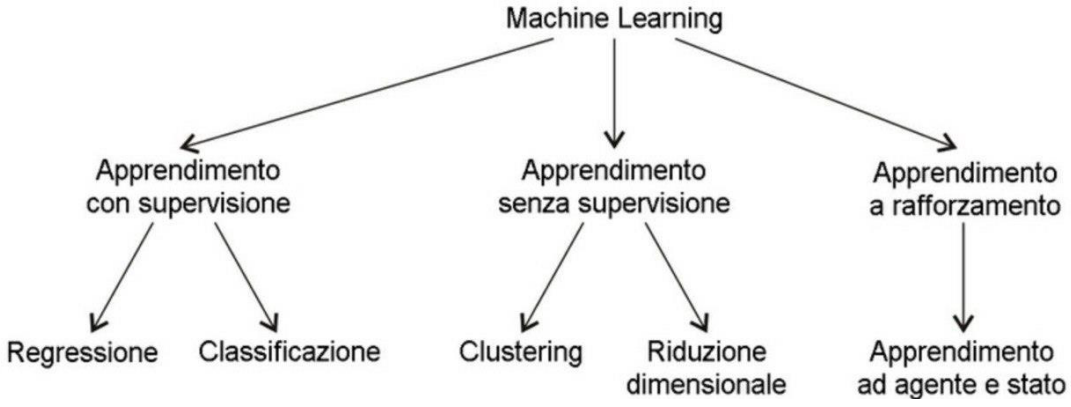


Figura 5.1. Schema modelli di apprendimento

In questo studio sono state effettuate analisi tramite algoritmi di apprendimento supervisionato, e nello specifico di classificazione. Tutti gli algoritmi di machine learning considerati hanno come denominatore comune la strategia di convalida e il

punto operativo di interesse definito sulla curva ROC in modo da poter confrontare agevolmente i vari modelli.
Nello specifico la strategia di convalida usata, nota come Hold Out, consiste nell'addestrare il modello su una parte di dati casuale (tipicamente un 80%) e testarne la sua qualità sulla rimanente parte (solitamente il 20%). Tale metodica è veloce in quanto il modello è valutato solamente una volta ma è molto sensibile alla varianza.
Tutto ciò è stato realizzato mediante la funzione "train_test_split" facente parte di "scikit-learn", libreria open source di apprendimento automatico progettata per interagire con le librerie numeriche e scientifiche Python NumPy e SciPy.
Invece per i punti operativi (meglio noti come "thresholds") è importante sottolineare che sono stati scelti in funzione del fatto di mantenere una sensitivity pari a 0.80.

5.1 Regressione logistica

La regressione logistica è un algoritmo di classificazione supervisionato che fornisce un risultato, che di fatto rappresenta una *probabilità* che un dato valore di ingresso appartenga a una determinata classe.
La regressione logistica è utile per effettuare analisi predittive, soprattutto per variabili binarie ovvero variabili che possono assumere solamente due valori 0 e 1. Solitamente il valore 1 rappresenta la classe positiva, mentre il valore 0 rappresenta la classe negativa.
L'obiettivo della regressione logistica è quello di misurare la relazione tra la variabile d'uscita y (variabile dipendente) e le possibili variabili d'ingresso x1, x2, ..., xn (variabili indipendenti) che possono essere numeriche o nominali.
Il modello non può essere definito con l'equazione di regressione lineare $p(X) = \beta_0 + \beta_1 X$, in quanto il valore della Y dato il set di predittori non può assumere un qualsiasi valore da meno infinito a più infinito, bensì deve necessariamente essere pari ad un valore compreso nell'intervallo [0,1] trattandosi di una probabilità.
Si ricorre pertanto alla trasformazione logistica tramite la funzione:

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

Dove $p(X) = Pr(Y = 1|X)$, β_0 chiamato intercetta e β_1 sono i coefficienti della regressione.

Ovviamente, nel caso di più di una variabile dipendente, la funzione viene generalizzata:

$$p(x) = \frac{e^{\beta_0 + \sum_{x=1}^{n} \beta_j x_j}}{1 + e^{\beta_0 + \beta_1 x}}$$

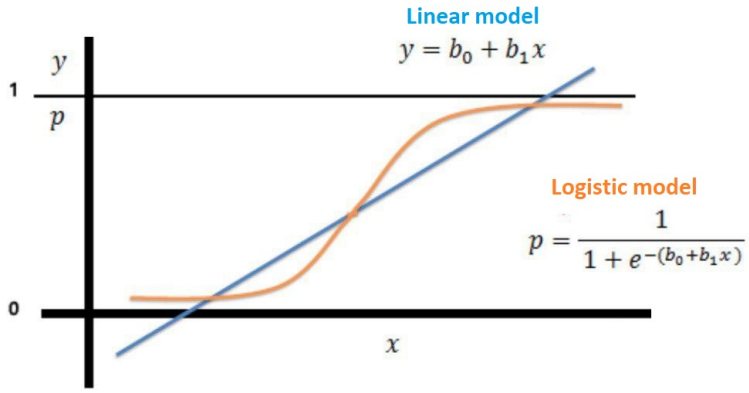


Figura 5.2. Modello lineare e modello logistico a confronto

Per spiegare gli elementi di base della regressione logistica come modello probabilistico, occorre innanzitutto introdurre il rapporto probabilistico (odds), ovvero il rapporto tra la probabilità di evento e la probabilità che lo stesso evento non accada.
Il rapporto probabilistico può essere scritto come:

$$odds = \frac{p}{1 - p}$$

dove p è la probabilità dell'evento positivo.
Il termine evento positivo non significa necessariamente buono, ma fa riferimento all'evento che vogliamo prevedere, per esempio la probabilità che un paziente sia affetto da una determinata malattia.
È possibile definire la regressione logistica come una regressione lineare fra la nostra caratteristica X e il log-odds che i nostri dati appartengano a una determinata

classe.

$$\begin{aligned}
\ln(odds) &= \ln\left[\frac{p(x)}{1-p(x)}\right] = \ln\left[\frac{\frac{e^{\beta 0+\beta 1^x}}{1+e^{\beta 0+\beta 1^x}}}{1-\frac{e^{\beta 0+\beta 1^x}}{1+e^{\beta 0+\beta 1^x}}}\right] \\
&= \ln\left[\frac{\frac{e^{\beta 0+\beta 1^x}}{1+e^{\beta 0+\beta 1^x}}}{\frac{1+e^{\beta 0+\beta 1^x}-e^{\beta 0+\beta 1^x}}{1+e^{\beta 0+\beta 1^x}}}\right] \\
&= \ln\left[\frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}\frac{1+e^{\beta_0+\beta_1 x}}{1}\right] \\
&= \ln\left[e^{\beta_0+\beta_1 x}\right] = \beta_0+\beta_1 x
\end{aligned} \tag{5.1}$$

Per la scelta dei coefficienti si utilizza il metodo della *massima verosimiglianza* che si basa sulla probabilità di osservare l'insieme dei dati in funzione dei β. Ottenere i coefficienti β_0 e β_1 non è immediato, in quanto si eseguono calcoli iterativi al fine di massimizzare la likelihood function:

$$l(\beta_0,\beta_1) = \prod_{i:y_\mathrm{i}=1} p(x_\mathrm{i}) \prod_{i:y_\mathrm{i}=0} 1-p(x_\mathrm{i}) \tag{5.2}$$

5.1.1 Risultati ottenuti

Prediction time	**Onset**	**12ore**	**24ore**	**48ore**
AUROC	0.89	0.84	0.80	0.76
Sensitivity	0.80	0.80	0.80	0.80
Specificity	0.91	0.71	0.59	0.50
Accuracy	0.90	0.72	0.61	0.53
DOR	42.97	9.94	5.70	3.91
LR+	9.39	2.79	1.94	1.58
LR-	0.22	0.28	0.34	0.40

Tabella 5.1. Risulati regressione logistica

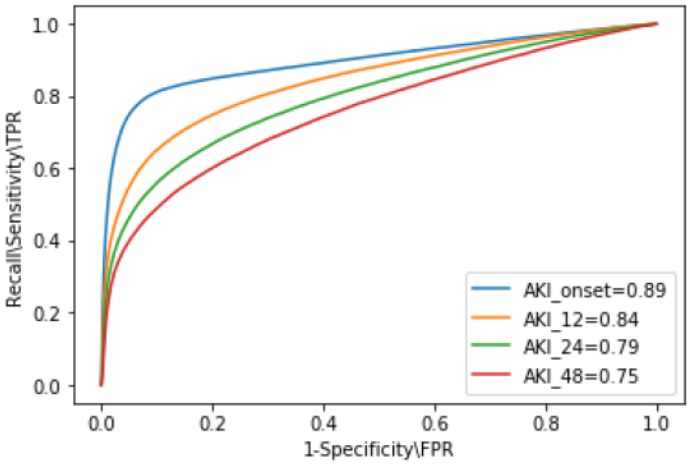


Figura 5.3. ROC

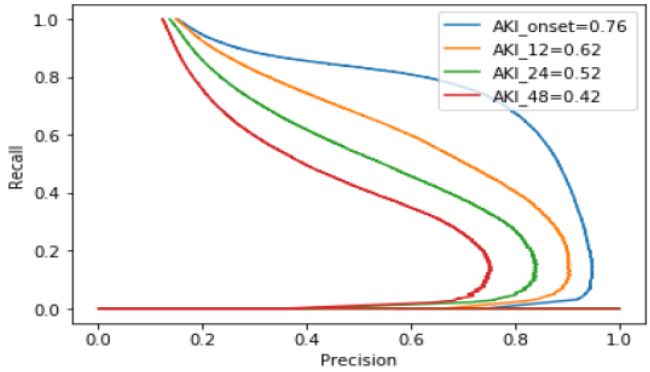


Figura 5.4. PRC

5.2 K-nearest neighbor classifier (KNN)

I classificatori instance-based anziché costruire modelli classificano nuovi record sulla base della loro similarità rispetto ai record appartenenti al training set.
Questi tipi di classificatori vengono detti lazy-learners in contrapposizione agli alberi decisionali, reti neurali ecc.
K-nearest neighbors (KNN) fa parte di questa famiglia di classificatori e viene detto "lazy" non per la sua apparente semplicità, ma piuttosto perchè non apprende una funzione di discriminazione dai dati di addestramento e memorizza quindi il dataset di addestramento. Vista la sua natura molto semplice, KNN può essere

schematizzato nei passi seguenti:

1. Scegliere il numero k e una metrica di distanza.
2. Trovare i k elementi più vicini del campione che si vuole classificare.
3. Assegnare l'etichetta della classe sulla base di un voto a maggioranza.

La Figura 5.5 mostra ciò che succede al variare del numero K di vicini scelti per l'implementazione dell'algoritmo.

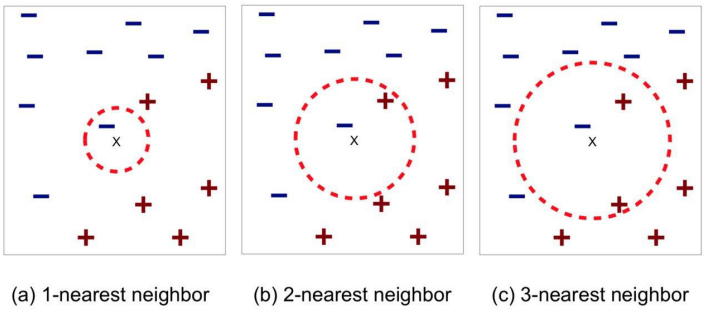


Figura 5.5. Scenari possibili al variare di k

I "k vicini più vicini" di un punto x sono i punti del training set che hanno le più piccole k distanze da x. La classificazione di un record z è ottenuta con un processo di voto a maggioranza tra i k elementi D_z del training set D più vicini (o simili) a z, secondo la seguente formula:

$$\bar{y} = \underset{y \in Y}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} I(y_i = y)$$

Y costituisce l'insieme delle etichette di classe. $I(y_i = y)$ restituisce 1 se il suo argomento è TRUE, 0 altrimenti.
La scelta di k è di fondamentale importanza in quanto se si scelgono valori di k molto piccoli si è sensibili al rumore, contrariamente per valori molto grandi di k l'intorno di punti può includere esempi appartenenti ad altre classi.
Inoltre, per operare in maniera corretta tutti gli attributi devono essere riportati alla medesima scala di valori, venendo quindi normalizzati in fase di pre-processing.
Di seguito vengono riassunti i punti di forza e le relative debolezze dell'algoritmo KNN.

Vantaggi:

- Questo approccio basato sulla memorizzazione fa si che il classificatore si adatti immediatamente mentre vengono raccolti nuovi dati di addestramento.
- Non richiede che venga costruito un modello.
- Rispetto a Rule-based system o decision tree, KNN permette di costruire "contorni" delle classi non lineari venendo considerato quindi molto flessibile.

Svantaggi:

- La complessità computazionale per la classificazione di nuovi campioni cresce in modo lineare con il numero di campioni presenti nel dataset di addestramento (nella situazione peggiore). Inoltre, non possono essere eliminati i campioni di addestramento, in quanto non esiste un passo di addestramento. Pertanto, lo spazio di memorizzazione può diventare un problema se ci si trova a lavorare con dataset di grandi dimensioni.
- È richiesta una misura di distanza o similarità.
- La classe di appartenenza di un nuovo punto viene determinata localmente e quindi può essere suscettibile al rumore dei dati.
- È richiesta una fase di pre-processing per la normalizzazione degli attributi.
- La presenza di attributi irrilevanti o correlati potrebbero falsare le distanze tra gli oggetti.

Nell'implementazione di k-nearest neighbors è stata scelta come metrica di distanza quella di "minkowski" vista semplicemente come una generalizzazione delle distanze euclidea e Manhattan.

$$d(x^{(i)}, x^{(j)}) = \sqrt[p]{\sum_k |x_k^{(i)} - x_k^{(j)}|^p}$$

In scikit-learn quest'ultima diviene una distanza euclidea impostando il parametro p=2. La Figura 5.6 mostra il tasso di errore al variare del parametro k. Come si evince dal grafico la scelta migliore è k uguale a 9.

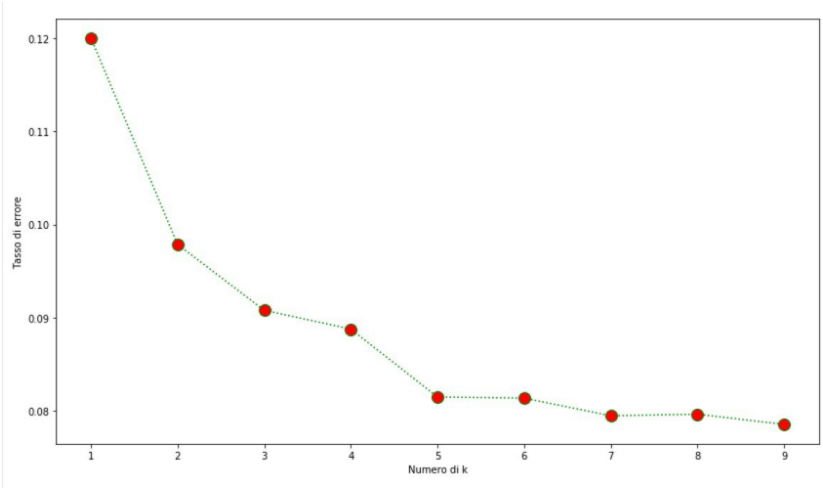


Figura 5.6. Scenari possibili al variare di k

5.2.1 Risultati ottenuti

Prediction time	Onset	12ore	24ore	48ore
AUROC	0.87	0.80	0.74	0.70
Sensitivity	0.78	0.80	0.76	0.74
Specificity	0.89	0.57	0.53	0.48
Accuracy	0.91	0.81	0.79	0.76
DOR	47.05	10.40	6.35	4.55
LR+	7.30	1.92	1.62	1.45
LR-	0.23	0.33	0.44	0.51

Tabella 5.2. Risulati KNN

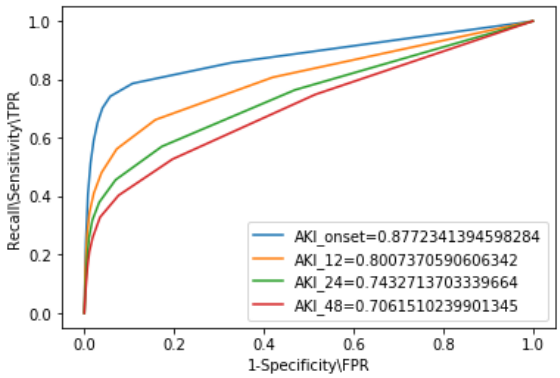


Figura 5.7. ROC

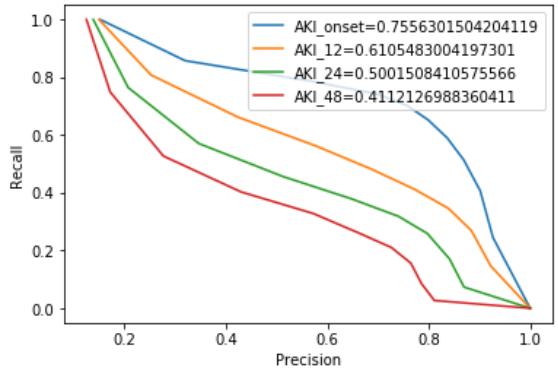


Figura 5.8. PRC

5.3 Random forest

Random forest ha raggiunto grande popolarità tra gli algoritmi di machine learning durante l'ultimo decennio in primis per le sue buone prestazioni come classificatore e in secondo luogo grazie alla sua scalabilità e facilità di utilizzo.
Come suggerisce lo stesso nome, si intuisce che le foreste casuali possono essere considerate come un insieme di alberi decisionali.
L'idea che sta alla base di random forest è la costruzione di un modello robusto mettendo insieme più sistemi di apprendimento deboli. Ogni albero genera una previsione sulla classe, e si prende come previsione del modello quella che ottiene il

maggior numero di voti (vedere la Figura 5.9) [27].

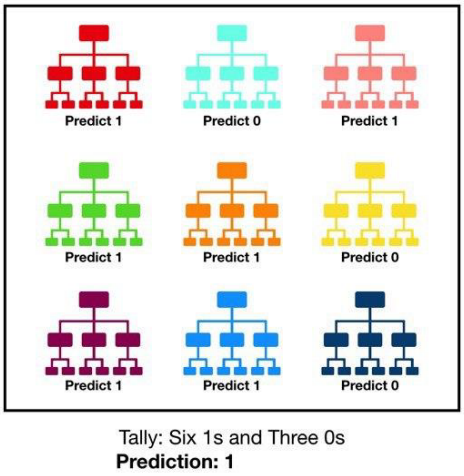


Figura 5.9. Foresta di alberi

L'algoritmo di random forest può essere riassunto nei seguenti passi:

1. Estrarre un campione casuale iniziale, noto come bootstrap, di dimensioni n (scegliere in modo casuale n campioni dal training set con reinserimento).

2. A partire dal campione di bootstrap far crescere un albero decisionale Per ogni nodo:

a. selezionare in modo casuale d caratteristiche senza reinserimento;

b. suddividere il nodo utilizzando la caratteristica che fornisce la migliore suddivisione sulla base della funzione obiettivo (per esempio massimizzando il guadagno informativo).

3. Ripetere per k volte i passi 1 e 2.

4. Mettere insieme le previsioni fatte da ciascun albero per assegnare l'etichetta della classe sulla base di un voto a maggioranza.

La Figura 5.10 mostra i passi sopra descritti. La bassa correlazione tra i modelli

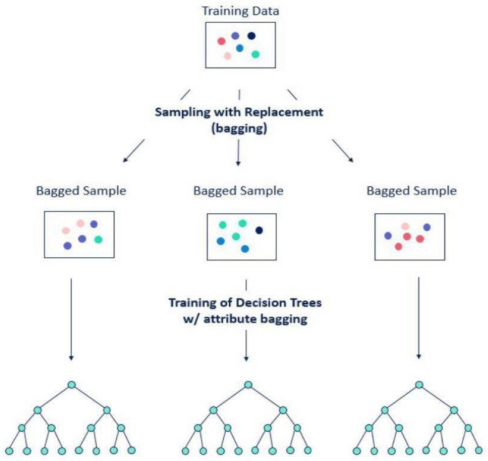


Figura 5.10. Algoritmo random forest

che formano random forest costituisce la chiave principale per produrre previsioni d'insieme molto più accurate di qualsiasi previsione individuale.
Un grande vantaggio di random forest è il fatto che non bisogna preoccuparsi troppo della scelta di un buon valore per gli iper-parametri.
L'unico parametro a cui bisogna prestare attenzione è il numero di alberi k scelti al passo 3. Tipicamente, maggiore è il numero di alberi, migliori saranno le prestazioni di random forest, con lo svantaggio però di un costo computazionale più elevato.
Tramite le dimensioni n del campione di bootstrap viene controllato il compresso bias-varianza. Scegliendo un valore elevato di n si riduce la casualità aumentando le probabilità di portare la foresta in overfitting.
Il livello di overfitting può essere ridotto scegliendo valori più piccoli di n, a discapito però di una riduzione delle prestazioni del modello in termini di velocità.
La maggioranza delle implementazioni, compresa l'implementazione RandomForestClassifier di scikit-learn, sceglie la dimensione del campione di bootstrap in modo tale da essere uguale al numero di campioni del set di addestramento originario, il che rappresenta un buon compromesso fra varianza e bias.
Per il numero di caratteristiche d a ciascuna suddivisione, è opportuno che si scelga un valore che sia più compatto del numero totale di caratteristiche presenti nel set

di addestramento. Un valore predefinito ragionevole che viene utilizzato in scikit-learn e altre implementazioni è $d = \sqrt{m}$, dove m è il numero di caratteristiche presenti nel set di addestramento.
Il risultato finale restituito dal random forest altro non è che la media del risultato numerico restituito dai diversi alberi nel caso di un problema di regressione, o la classe restituita dal maggior numero di alberi nel caso in venga utilizzato per problemi di classificazione.

5.3.1 Risultati ottenuti

Prediction time	**Onset**	**12ore**	**24ore**	**48ore**
AUROC	0.89	0.82	0.77	0.73
Sensitivity	0.80	0.80	0.78	0.79
Specificity	0.88	0.66	0.54	0.45
Accuracy	0.87	0.68	0.57	0.49
DOR	31.84	7.82	4.39	3.20
LR+	7.16	2.36	1.71	1.45
LR-	0.22	0.30	0.39	0.45

Tabella 5.3. Risulati random forest

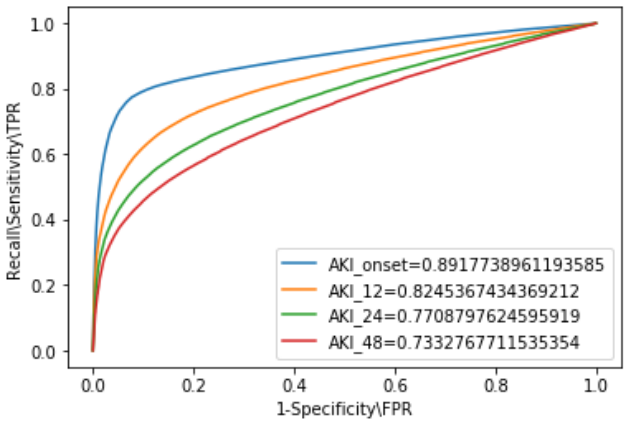


Figura 5.11. ROC

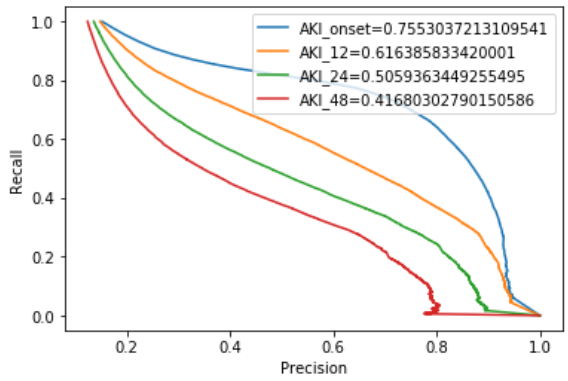


Figura 5.12. PRC

5.4 XGBoost

Extreme Gradient Boosting (XGBoost) è uno strumento di machine learning molto flessibile, efficiente e versatile che può risolvere la maggior parte dei problemi di regressione, classificazione.
XGBoost è stato sviluppato nel 2016 da Tianqi Chen e successivamente è diventato popolare per aver vinto numerose competizioni Kaggle. XGBoost ha dimostrato di spingere i limiti della potenza di calcolo per gli algoritmi degli alberi potenziati poiché è stato costruito e sviluppato al solo scopo di prestazioni del modello e velocità di calcolo.
XGBoost rappresenta l'evoluzione degli alberi decisionali, come mostrato in Figura 5.13, ed è un'implementazione dell'algoritmo degli alberi di gradient boosting.

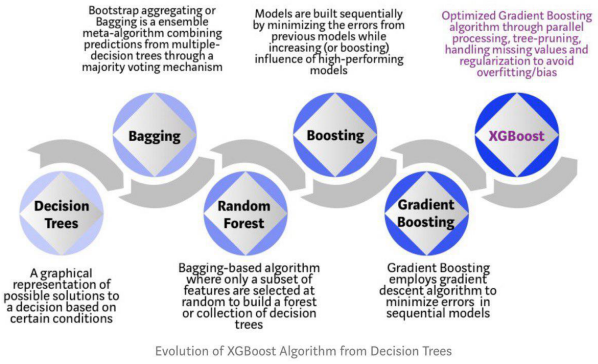


Figura 5.13. Evoluzione XGBoost

Il boosting (potenziamento) è un metodo ensemble, ovvero presuppone che i modelli siano costruiti in modo iterativo cosicchè ogni modello successivo possa ridurre gli errori del modello precedente. Nello specifico, ogni albero costruito impara dai suoi precedessori e aggiorna gli errori residui fino a quando i dati di addestramento non vengono accuratamente previsti o riprodotti dal modello.
Il gradient boosting è un algoritmo che utilizza la tecnica del boosting ma considera la discesa del gradiente per ridurre al minimo la perdita quando si aggiungono nuovi modelli, in quanto si rende questo processo generico e applicabile a tutte le funzioni di perdita. La discesa del gradiente permette di minimizzare qualsiasi funzione differenziabile. Ai fini di addestramento del modello è necessario definire la funzione obiettivo che serve a misurare il modo in cui il modello si adatta ai dati. In XGBoost la discesa del gradiente viene utilizzata per ottimizzare la funzione obiettivo.
La funzione obiettivo è definita come:

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

dove L è la *funzione di perdita*, misura quanto è predittivo il modello rispetto ai dati di allenamento. Tipicamente vengono usate:
• Rooted Mean Squared Error per regressione
$L = \frac{1}{N}\sum_{i=1}^{N}(y_\mathrm{i} - \hat{y}_\mathrm{i})^2$
• LogLoss per classificazione binaria
$L = -\frac{1}{N}\sum_{i=1}^{N} y_\mathrm{i} log(p_\mathrm{i}) + (1 - y_\mathrm{i}) log(1 - p_\mathrm{i})$
• Mlogloss per multi-classificazione
$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_\mathrm{i,j} log(p_\mathrm{i,j})$

Invece, Ω definisce il termine di regolarizzazione e permette di controllare la complessità del modello evitando un eccessivo adattamento (overfitting). La *funzione di regolarizzazione* è definita come:
$\Omega = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$
Dove w è il vettore dei punteggi sulle foglie, T è il numero di foglie, λ è il coefficiente di regolarizzazione $\lambda > 0$ introdotto per bilanciare i due termini della funzione obiettivo, γ è il learning rate.

Dal punto di vista del sistema, la libreria consente funzionalità come la parallelizzazione per la costruzione di alberi su più core della CPU durante la fase di

training, calcolo distribuito per l'addestramento di modelli di grandi dimensioni, elaborazione out-of-core per dataset molto grandi che non si adattano alla memoria principale e ottimizzazione della cache per migliorare l'utilizzo e l'efficienza dell'hardware. Tutto questo permette all'algoritmo di ridurre in modo efficiente i tempi di elaborazione e di avere un utilizzo ottimale delle risorse di sistema.
Inoltre, XGBoost offre un'ampia varietà di parametri di ottimizzazione ed è in grado di gestire internamente i valori mancanti nel caso in cui fossero presenti nel set di dati
Per la configurazione di XGBoost sono stati settati i seguenti parametri nel rispettivo modo:
Parametri generali
• nthread = 10 Numero di thread paralleli utilizzati per eseguire XGBoost.
• max_depth = [Default = 6] Profondità massima di un albero. L'aumento di questo valore renderà il modello più complesso e con maggiori probabilità di sovralimentarsi.
• Eta = [impostazione predefinita = 0,3, alias: learning_rate] Rappresenta il restringimento della dimensione del passo utilizzato nell'aggiornamento per evitare un adattamento eccessivo.
Parametri d'apprendimento
• objective = [binary:logistic] Regressione logistica per classificazione binaria, probabilità di uscita.
• eval_metric = ['auc','error'] Metriche di valutazione per i dati di convalida. auc: Area sotto la curva, error: Tasso di errore di classificazione binaria, definito come (wrong cases) / (all cases).
Per le previsioni, la valutazione considererà le istanze con un valore di previsione maggiore di 0,5 come istanze positive e le altre come istanze negative.

5.4.1 Risultati ottenuti

Prediction time	Onset	12ore	24ore	48ore
AUROC	0.90	0.84	0.79	0.76
Sensitivity	0.80	0.80	0.80	0.80
Specificity	0.91	0.71	0.58	0.50
Accuracy	0.89	0.73	0.61	0.53
DOR	43.22	10.23	5.59	4.04
LR+	9.44	2.84	1.91	1.60
LR-	0.21	0.27	0.34	0.39

Tabella 5.4. Risulati Xgboost

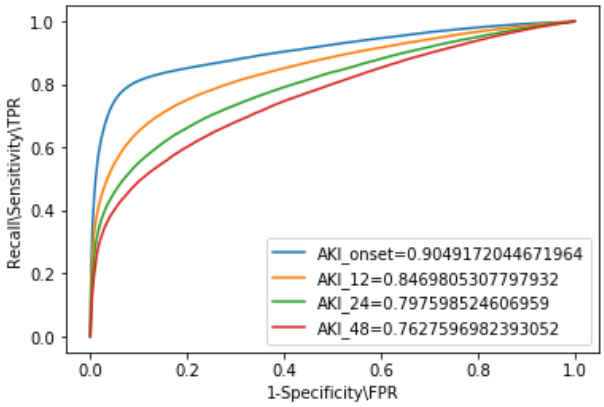


Figura 5.14. ROC

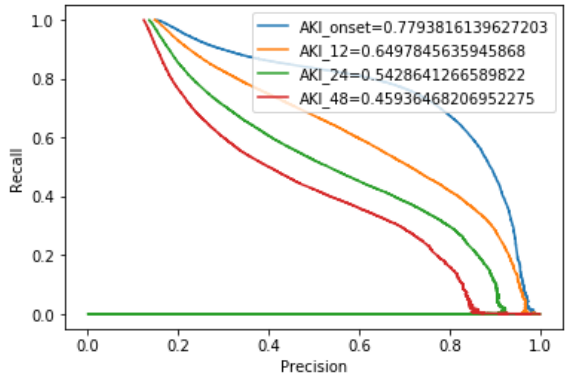


Figura 5.15. PRC

5.5 Introduzione storica alle reti neurali

La nascita delle reti neurali ha inizio nei primi anni '40, quando due scienziati di nome Warren Sturgis McCulloch e Walter Pitts, rispettivamente neurofisiologo e matematico, in un famoso lavoro del 1943 intitolato "A Logical Calculus of the Ideas Immanent in Nervous Activity" mostrarono che il neurone fosse l'unità logica di base del cervello cercando di dimostrare che una macchina di Turing poteva essere realizzata con una rete finita di neuroni.
I due scienziati modellarono il primo neurone artificiale identificandolo come un "combinatore lineare a soglia", ovvero un sistema in cui per lo strato di input erano previsti dati binari multipli in entrata e per quello di output un singolo valore binario in uscita.
Si dimostrò che che la rete neurale, creata combinando questi elementi, era in grado di calcolare funzioni booleane molto semplici. Fino alla fine degli anni '50 non ci furono clamorose scoperte.
Nel 1949 Donald Olding Hebb, psicologo canadese, cercò di spiegare i modelli complessi del cervello formulando le prime ipotesi di apprendimento delle reti neurali.
Nel 1958 invece Frank Rosenblatt, psicologo e computer scientist americano, presentò il primo schema di rete neurale chiamandolo Perceptron. Quest'ultimo era formato da uno strato di ingresso e uno di uscita ed una regola di apprendimento intermedia alla cui base vi era l'algoritmo di 'error back-propagation' che alterava i pesi nei collegamenti tra i neuroni in funzione della differenza tra l'output ottenuto e quello desiderato.

Per circa un decennio le teorie di Ronsenblatt animarono il mondo scientifico, fintanto che nel 1969 Marvin Minsky e Seymour Papert evidenziarono i limiti del Percettrone nel riconoscere solamente funzioni linearmente separabili, incapace quindi di calcolare la funzione "OR esclusivo", meglio nota come "XOR". L'attrazione per la computazione neurale quindi venne accantonata per un altro decennio.
Nel 1986 grazie a David Rumelhart fu introdotto il terzo strato delle reti neurali, chiamato hidden layer, che caratterizza il sistema di addestramento delle reti Multi-Layers Perceptron (MLP). Sempre a Rumelhart si deve l'algoritmo di retropropagazione dell'errore (error backpropagation) grazie al quale i pesi delle connessioni tra i nodi vengono modificati sistematicamente in modo tale da far avvicinare sempre di più la risposta della rete a quella desiderata. Tale algoritmo è oggi usato nell'apprendimento supervisionato.
La scoperta di algoritmi di apprendimento basati sulla backpropagation e la disponibilità di elaboratori molto più potenti rispetto ai decenni prima, è stato decisivo nel far esplodere nuovamente negli anni '90 l'interesse della comunità scientifica per quanto concerne la computazione neurale.

5.6 Cosa sono le reti neurali artificiali

Le reti neurali artificiali sono modelli matematici formati da neuroni artificiali di ispirazione alle reti neurali biologiche (quella umana o animale) e vengono utilizzate per risolvere problemi ingegneristici di Intelligenza Artificiale legati a diversi ambiti tecnologici come l'informatica, l'elettronica, la simulazione o altre discipline. Le reti neurali artificiali sono modelli di calcolo di natura matematica-informatica il cui funzionamento è ispirato alle reti neurali biologiche, ovvero modelli costituiti da informazioni interconnesse fra di loro. Queste interconnessioni si basano su processi di calcolo chiamati Parallel Distributed Processing (PDP).
Il cervello umano elabora le informazioni sensoriali in maniera parallela e le distribuisce ai vari nodi della rete. Nell'informatica tradizionale i calcoli avvengono in maniera seriale (e non in parallelo) e vi è una memoria centrale in cui i dati vengono memorizzati. Basandosi su informazioni provenienti dall'esterno e dall'interno che si connettono attraversando la rete nella fase di apprendimento, una rete neurale può modificare i suoi nodi e le sue interconnessioni, quindi la sua struttura.
Per comprendere al meglio il funzionamento di una rete neurale artificiale è opportuno descrivere ad alto livello il funzionamento di una rete neurale biologica.

Una rete neurale biologica riceve segnali dall'esterno (ad esempio negli uomini e negli animali essi vengono percepiti tramiti i sensi grazie a cellule nervose molto complesse aventi diversi task come riconoscere gli stimoli o percepire l'ambiente) che vengono processati e quindi elaborati in informazioni attraverso una grande quantità di neuroni (strutture in cui avvengono i calcoli) connessi fra loro in una struttura non lineare che varia in funzione di stimoli esterni. Le reti neurali artificiali ricevono dall'esterno dei segnali su uno strato di nodi (rappresentante l'unità di elaborazione), quest'ultimi sono connessi a molteplici nodi interni alla rete organizzati in più livelli.
Schematizzando la struttura di una rete neurale, si possono individuare tre strati principali caratterizzati da migliaia di neuroni e decine di migliaia di connessioni: Input, Hidden e Output. Il compito principale dello strato di input è di processare i dati ricevuti in ingresso e riuscirli a preparare alle richieste dei neuroni. Il compito di elaborazione delle informazioni è demandato allo strato di hidden (noto come strato nascosto) che può essere formato da uno o più livelli di neuroni. Infine, lo strato di output raccoglie i risultati provenienti dallo strato di hidden e li adatta alle richieste del livello successivo.
I campi di applicazione di una rete neurale possono essere processi di ottimizzazione, memorie associative, elaborazione di segnali ed immagini, pattern recognition, clustering, approssimazione di funzioni e predizioni in serie temporali.
Una rete neurale va addestrata a risolvere certe tipologie di problemi, quindi occorre una fase di apprendimento in cui insegnarle come comportarsi quando riceve un certo input dato che all'inizio essa non possiede nessuna forma di conoscenza.
I modelli di rete neurale si differenziano anche per il modo in cui avviene l'apprendimento dato che ogni modello possiede il suo tipo e non può usarne altri al fine di apprendere. Apprendimento supervisionato, non supervisionato, per rinforzo e infine semi-supervisionato costituiscono i modelli tramite cui una rete neurale artificiale può apprendere informazioni in modo automatico e adattativo.
Nell'apprendimento supervisionato, noto come supervised learning, l'algoritmo ha a disposizione sia gli input che i dati che si vorrebbero ottenere al fine di far identificare alla rete stessa una qualche regola generale che colleghi i dati in ingresso con quelli di uscita. La rete quindi ha il compito di imparare un nesso tra di essi in modo tale da estrarre una regola che possa essere riutilizzata in problemi simili. Perceptron e multilayer perceptron sono esempi di reti che usano questo modello di apprendimento.

Nell'apprendimento non supervisionato (unsupervised learning) alla rete vengono forniti dati senza specificare il risultato che si vorrebbe ottenere. Ciò ha lo scopo di riuscire a risalire a pattern nascosti ovvero a identificare negli input una certa logica anche se quest'ultimi non possiedono un'etichetta di classe.
Nell'apprendimento per rinforzo la rete raggiunge un obiettivo e riceve una ricompensa interagendo con un ambiente dinamico, e riesce anche ad imparare dagli errori identificati mediante "punizioni". Quindi alla base di questo apprendimento vi è uno schema di ricompensa/punizione.
L'apprendimento semi-supervisionato è un ibrido in cui viene fornito un insieme di dati incompleti nella fase di training, alcuni dei quali possiedono le etichette di classi mentre altri ne sono privi.

5.7 Perceptron

La forma più semplice di rete neurale è rappresentata dal singolo perceptron. Prima di descrivere quindi l'architettura del multilayer perceptron e presentare i relativi risultati in questo studio, è opportuno fornire una breve panoramica sul funzionamento del perceptron su cui si basa MLP. Come già detto nel paragrafo 5.5 il perceptron fu presentato da Rosenblatt nel 1958.
Il perceptron è un classificatore lineare (binario) dato che viene solitamente utilizzato per classificare i dati in due parti, come mostrato in Figura 5.16, ed è utilizzato nell'ambito dell'apprendimento con supervisione.

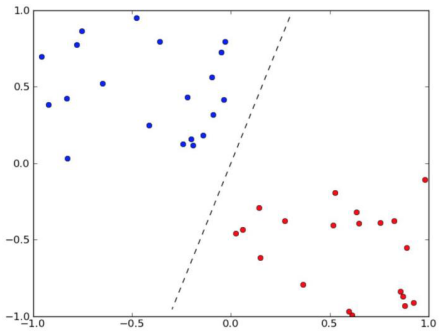


Figura 5.16. Classificazione lineare

Il perceptron è composto da quattro parti come mostrato in Figura 5.17:

- strato di input
- pesi e bias
- sommatoria
- funzione di attivazione

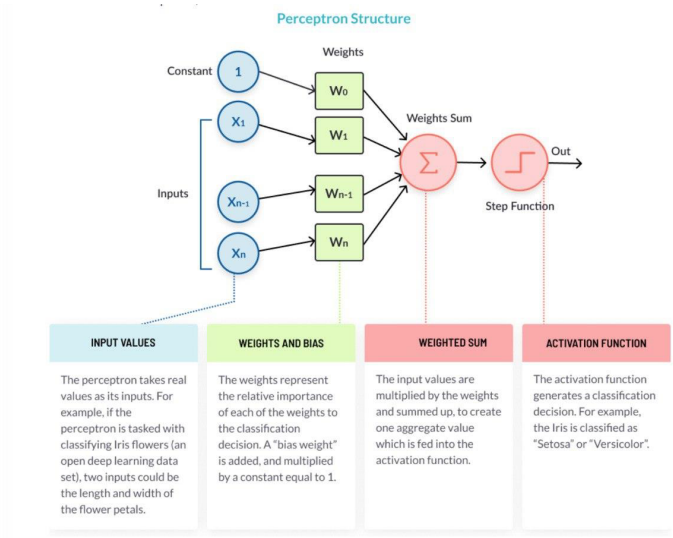


Figura 5.17. Struttura del perceptron

Un perceptron accetta un certo input e produce un segnale come output. Questo segnale viene ottenuto combinando l'input con vari pesi e poi attraversando una funzione di attivazione. Nel caso di semplici output binari, in genere si usa la funzione logistica: $f_{\log}(z) = \frac{1}{1+e^{-z}}$

I pesi servono a mostrare il contributo di un nodo in particolare. Il bias consente di spostare la curva della funzione di attivazione su o giù. Una funzione di attivazione serve a mappare l'input tra i valori richiesti come ad esempio nei range (0, 1) o (-1, 1).
Il suo semplice funzionamento può essere riassunto nei seguenti passi:

1. Gli ingressi X vengono moltiplicati per i loro pesi W. K indicherà questo prodotto.
2. Fare la somma pesata di tutti i valori moltiplicati.

3. Applicare quella somma ponderata alla corretta funzione di attivazione.

La Figura 5.18 mostra un esempio di funzione di attivazione.

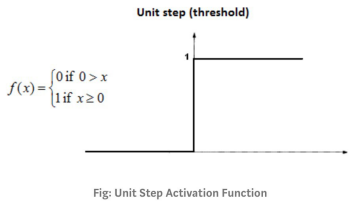


Figura 5.18. Funzione di attivazione

La Figura 5.19 mostra una situazione in cui il Perceptron riesce a trovare una retta per separare le due classi di oggetti.
Mentre la Figura 5.20 mette a risalto il limite principale del Perceptron rappresentato dal fatto di non trovare una retta di separazione.

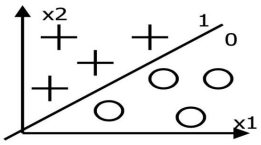


Figura 5.19. Separazione di due classi con perceptron

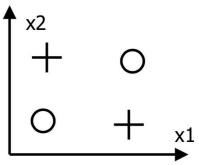


Figura 5.20. Limite del perceptron

5.8 Multilayer perceptron (MLP)

Le limitazioni introdotte dal percetron vengono superate da MLP. Quest'ultimo, aggiungendo alla sua architettura degli strati di neuroni nascosti (chiamati hidden layer) rappresenta internamente gli input in modo più complesso individuando regioni arbitrarie riuscendo a intersecare iperpiani nello iperspazio dei valori input.

Per creare una rete neurale, bisogna collegare fra loro più perceptron. Quindi MLP è visto come un grafo finito aciclico. I nodi sono neuroni con funzione d'attivazione logistica. La Figura 5.21 mostra un esempio di architettura di MLP.

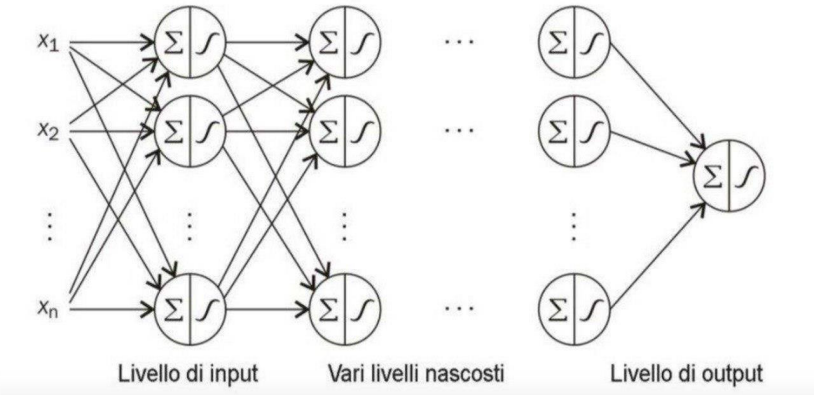


Figura 5.21. Architettura MLP

Mentre si addestra il modello, si aggiornano i pesi che all'inizio sono casuali in maniera tale da ottenere la migliore previsione possibile. Se dopo aver attraversato il modello un'osservazione produce un output falso quando invece sarebbe dovuto essere vero, le funzioni logistiche dei singoli perceptron vengono modificate leggermente. Tale tecnica è chiamata propagazione all'indietro (back-propagation).

Esempio:
Un perceptron formato da d input avrà come output:

$$y_{\mathrm{j}} = y\left(\sum_{i=0}^{d} w_{\mathrm{ji}} x_{\mathrm{i}}\right) \tag{5.3}$$

dove x_{i} sono gli input e w_{ji} sono i pesi di ogni input combinati con ogni output.
Adesso invece se si prende in considerazione una rete formata da due livelli di unità di elaborazione, con un numero di ingressi pari a d e con un numero di uscite pari a m per il primo strato e c uscite al secondo livello, gli output finali della rete saranno calcolati con la seguente formula:

$$z_{\mathrm{k}} = z\left(\sum_{j=0}^{m} w\prime_{\mathrm{kj}} y_{\mathrm{j}}\right) \tag{5.4}$$

dove z_k è l'output finale, $w\prime_{kj}$ sono i pesi di ogni unità di elaborazione, y_j è il segnale che viene inviato dalle unità nascoste.
Il bias è presente come coefficiente rispettivamente dell'ingresso x_0 e y_1 posti uguali a 1. Esso ha molta importanza in quanto assicura che la mappa della rete non sarà lineare.
Unendo quindi le equazioni (5.3) e (5.4) si arriva al risultato finale:

$$z_k = z\left(\sum_{j=0}^{m} w\prime_{kj} \quad y\left(\sum_{i=0}^{d} w_{ji}x_i\right)\right) \tag{5.5}$$

Le reti neurali vengono normalmente addestrate in lotti. Alla rete vengono forniti contemporaneamente vari punti dei dati di addestramento per più volte, e ogni volta l'algoritmo di back-propagation richiederà una modifica dei pesi interni della rete.
In uno scenario del genere si intuisce facilmente che una rete può crescere molto in profondità riuscendo ad avere molti livelli nascosti, che determinano la complessità della rete stessa. Quando le reti neurali crescono diventando molto profonde (formate da molti livelli), si entra nel campo dell'apprendimento profondo.
Le reti neurali profonde hanno come vantaggio principale il fatto di riuscire ad approssimare quasi ogni funzione e, teoricamente, possono apprendere le combinazioni ottimali di caratteristiche per poi usarle al fine di ottenere il massimo potere predittivo possibile.

Per la configurazione di MLP in questa tesi, dopo vari tentativi, si è constatato che le prestazioni migliori si hanno quando si settano i seguenti parametri nel rispettivo modo:
• max_iter=500 (Per iterazione si intende la presentazione dei pattern costituenti un mini-batch e il conseguentemente aggiornamento dei pesi)
• batch_size:default 'auto', quando si setta "auto", batch_size=min(200, n_samples)
• 2 hidden layer di cui il primo con 20 neuroni e il secondo con 6
• funzione di uscita soft_max
(Quando una rete neurale è utilizzata come classificatore multi- classe, l'impiego della funzione di attivazione softmax consente di trasformare i valori di net prodotti dall'ultimo livello della rete in probabilità delle classi.)

5.8.1 Risultati ottenuti

Prediction time	**Onset**	**12ore**	**24ore**	**48ore**
AUROC	0.90	0.84	0.79	0.76
Sensitivity	0.80	0.80	0.80	0.80
Specificity	0.92	0.73	0.58	0.50
Accuracy	0.90	0.74	0.61	0.54
DOR	45.11	10.72	5.56	4.04
LR+	9.82	2.94	1.91	1.61
LR-	0.22	0.27	0.34	0.40

Tabella 5.5. Risulati MLP

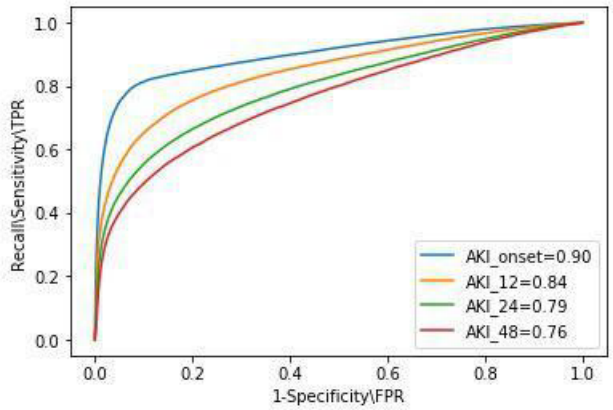


Figura 5.22. ROC

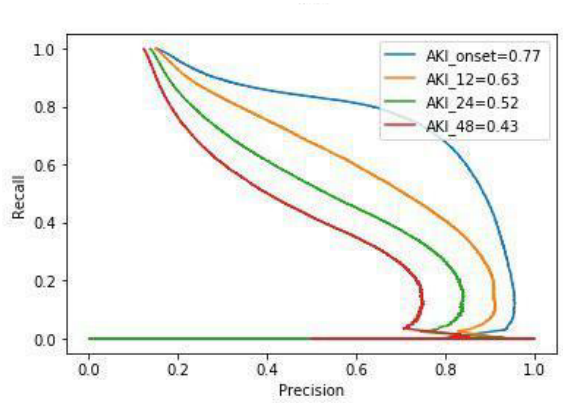


Figura 5.23. PRC

5.9 Reti neurali ricorrenti (RNN)

Gli umani non iniziano a pensare da zero ogni secondo. Mentre si legge questa tesi, si capisce ogni singola parola grazie alla comprensione di quelle precedenti. Non si ricomincia a pensare da zero buttando via tutto dato che i nostri pensieri hanno una certa perseveranza. Le reti neurali tradizionali non possono farlo e sembra una grande mancanza.
Vengono introdotte allora le reti neurali ricorrenti (recurrent neural network) che riescono a risolvere questo problema e si presentano come reti con loop interni (Figura 5.24), grazie ai quali le informazioni riescono a persistere. La presenza di cicli incide profondamente sulle capacità di apprendimento della rete e le relative performance, rendendo il sistema particolarmente dinamico.

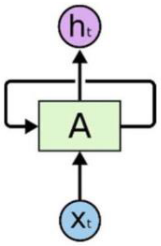


Figura 5.24. Rete con loop

Una di rete neurale di tale tipo, A, esamina alcuni input x_t e fornisce in uscita un valore h_t. Un loop consente di passare le informazioni da uno step della rete a quello successivo. Una rete neurale ricorrente può essere immaginata come più

copie della stessa rete, ognuna delle quali trasmette un messaggio a un successore. La Figura 5.25 mostra il loop srotolato:

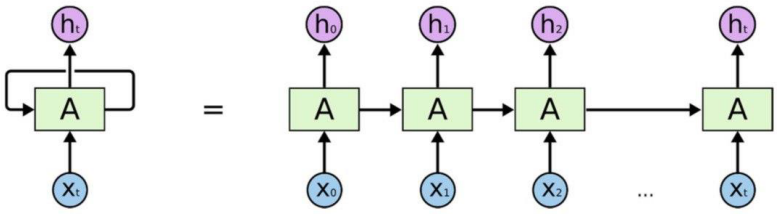


Figura 5.25. Rete con loop srotolato

Tutte le reti neurali ricorrenti hanno la forma di una catena di moduli ripetitivi di rete neurale. Nelle RNN standard, questo modulo ripetuto avrà una struttura molto semplice, come un singolo strato di tanh.

Una recurrent neural network (RNN) tiene traccia del passato e prende decisioni

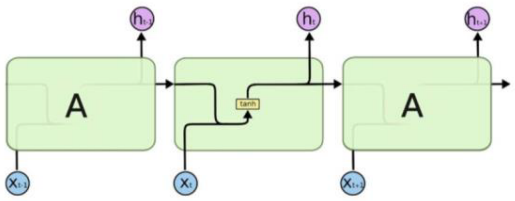


Figura 5.26. RNN

in base a ciò che ha imparato in quest'ultimo.

Se da un lato le reti feed forward ricordano ciò che hanno imparato durante la fase di training (per esempio un classificatore di immagini apprende l'aspetto di un "1" durante l'allenamento e quindi utilizza tale conoscenza per classificare altro), dall'altro lato le reti neurali ricorrenti imparano in modo simile durante l'allenamento, ma hanno anche il grosso vantaggio di ricordare le cose acquisite da precedenti input. Le reti neurali ricorrenti inoltre possono ricevere più vettori di input e produrre uno o più vettori di output.

Gli output sono influenzati oltre che dai pesi anche da un vettore di stato "nascosto" che rappresenta il contesto basato su precedenti ingressi/uscite. Quindi uno stesso input potrebbe produrre un output diverso a seconda degli input precedenti. Una RNN può essere usata anche per problemi legati ad immagini, che possono

essere scomposte in una serie di patch e trattate come una sequenza.

In conclusione, la Figura 5.27 mostra una semplice rete ricorrente proposta da Elman, in cui il BTSXPE nella parte inferiore dell'immagine rappresenta l'esempio di input nel momento corrente e CONTEXT UNIT rappresenta l'output del momento precedente.

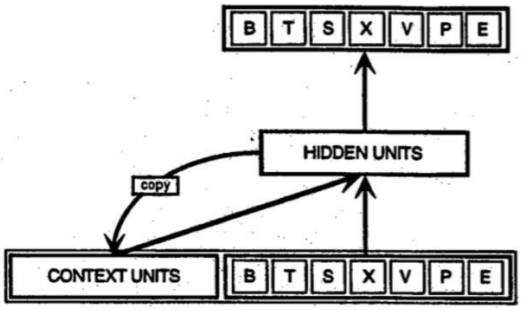


Figura 5.27. Esempio di RNN

Se una rete neurale ricorrente al tempo t-1 raggiunge una determinata decisione, quest'ultima influenzerà la decisione che sarà presa nella fase temporale successiva t.
Si può affermare quindi che le reti neurali ricorrenti hanno due fonti di input, ovvero presente e passato recente, che si uniscono al fine di rispondere ai nuovi input proprio come avviene nella vita di tutti i giorni.
Una RNN si distingue da una rete feedforward per un circuito di feedback collegato a decisioni passate e processa i propri output volta per volta così come degli input. Per questo motivo si dice che una rete di questo tipo goda di una "memoria" utile a eseguire attività su informazioni nella sequenza stessa, operazione che una rete feedforward non riesce a compiere.
Nello stato nascosto della rete ricorrente vengono memorizzate tutte quelle informazioni utili per elaborare nuovi esempi. Proprio come la memoria umana circola invisibilmente all'interno di un corpo influenzandone il comportamento, in una RNN l'informazione circola nei suoi stati nascosti.

5.10 Long short-term memory (LSTM)

A metà degli anni '90 due ricercatori tedeschi, Hochreiter e Schmidhuber, hanno introdotto una variante di rete neurale ricorrente nota come long short term memory (LSTM).
Ciò che distingue le RNN e LSTM da altre tipologie di reti neurali è che esse prendono in considerazione il tempo e la sequenza, ovvero vi è una dimensione temporale. Ciò che distingue invece una LSTM da una RNN è il tipo di operazioni che avvengono dentro ogni cella.

Una LSTM ha dei meccanismi interni chiamati "Gates" che hanno il compito di

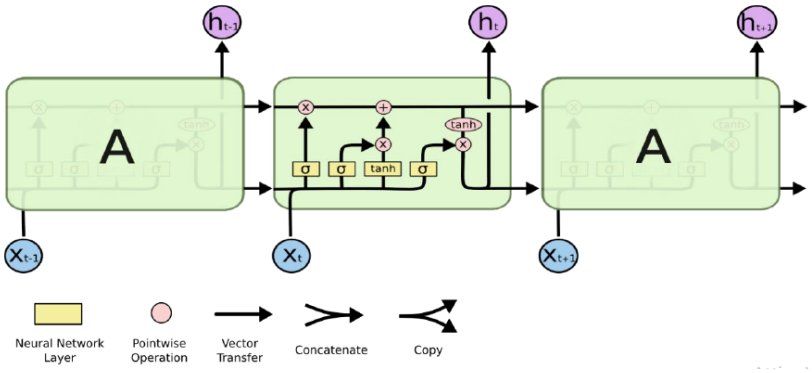


Figura 5.28. LSTM

regolare il flusso di informazioni. Questi Gates apprendono da una sequenza di dati quali sono i più importanti e quindi da conservare o quali eliminare. Questo meccanismo fa si che le informazioni più pertinenti possano attraversare tutta la catena di celle.
Una LSTM può essere impiegata nel riconoscimento e nella sintesi vocale, nella generazione di testi e nella generazione di didascalie per i video.
Il concetto chiave di LSTM è rappresentato da uno "stato", caratterizzante una cella, che agisce come un nastro trasportatore con lo scopo di trasferire le informazioni di interesse lungo tutta la rete. Lo stato della cella per tal motivo può essere quindi visto come "memoria" della rete.
I gates ricoprono un ruolo di estrema importanza in quanto è compito loro aggiungere o rimuovere le informazioni allo stato della cella durante la fase di training. I gates hanno al loro interno funzioni di attivazione sigmoidali e di tangente iperbolica. Un'attivazione sigmoidea è simile all'attivazione tanh ma invece di compattare

i valori tra -1 e 1, li compatta tra 0 e 1. In questo modo è possibile mantenere o dimenticare le informazioni.
Ad esempio, nel caso di informazioni inutili esse saranno moltiplicate per 0 e quindi scompariranno, mentre nel caso in cui vengano moltiplicate per 1 saranno conservate. Un'attivazione tanh invece permette la regolazione dei valori che passano attraverso la rete in modo che siano sempre compresi tra -1 e 1.

Entrando più nel dettaglio, ogni cella di LSTM al suo interno ha tre gates, ognuno dei quali ricopre un task diverso:

- **Forget Gate** : questo gate ha il compito di rimuovere le informazioni che non vengono più ritenute utili nello stato di cella C_t.Due ingressi x_t (input in un determinato momento) e h_{t-1} (precedente output di cella) vengono inviati al gate e moltiplicati per matrici di peso. Il risultato di questa operazione passa attraverso una funzione di attivazione che fornisce un output binario. Il risultato ottenuto dopo aver applicato la funzione di attivazione f_t viene moltiplicato con lo stato della cella precedente C_{t-1} , in modo da ottenere lo stato attuale della cella C_t.
  Se un particolare stato di cella ha come output 0, allora l'informazione viene dimenticata. Contrariamente per output uguale a 1, l'informazione viene conservata perchè ritenuta utile per usi futuri.

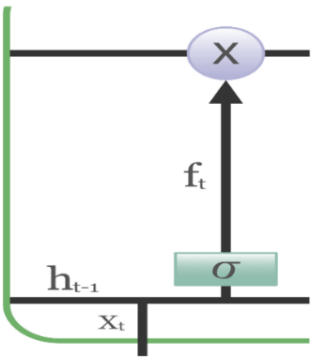


Figura 5.29. Forget gate

- **Input gate** : ha il compito di aggiungere informazioni utili allo stato della cella. Quest'ultime vengono regolate tramite l'utilizzo della funzione sigmoide

e vengono filtrati i valori da ricordare in modo simile al Forget gate usando gli ingressi h_{t-1} e x_t .Si crea un vettore utilizzando la funzione tanh che da un output da -1 a $+1$, e contiene tutti i possibili valori da h_{t-1} e x_t . Infine, i valori del vettore e i valori regolati vengono moltiplicati per ottenere informazioni utili.

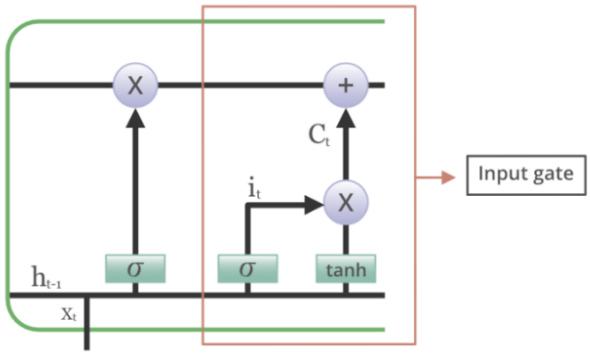


Figura 5.30. Input gate

- **Output gate** : estrae informazioni utili dallo stato di cella corrente da presentare come output. Si genera un vettore applicando la funzione tanh sulla cella, in questo modo le informazioni vengono regolate usando la funzione sigmoide e si filtrano i valori da ricordare tramite gli input h_{t-1} e x_t . Infine, i valori del vettore e i valori regolati vengono moltiplicati per essere inviati come output e passati come input alla cella successiva.

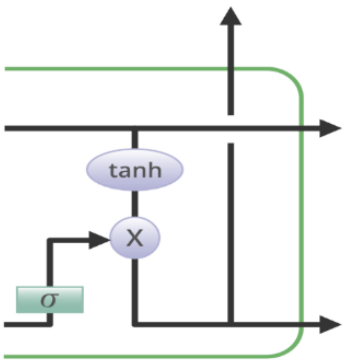


Figura 5.31. Output gate

Per la configurazione di LSTM si sono percorse due strade differenti sulla base dei hidden layer della rete. Si è considerata una rete con un hidden layer avente 25 neuroni e in segutio una rete a due hidden layer rispettivamente con 20 e 9 neuroni. In entrambe le configurazioni si è resa casuale la sequenza dei record del dataset, si sono normalizzati i valori di input e variati i valori di BATCH_SIZE e NUMERO DI EPOCHE.
Alla fine, si è constatato che le prestazioni migliori si hanno con la rete neurale a due livelli nascosti, con batch_size = (train_set) − (validation_set), numero di epoche = 1200.
Ai fini di una classificazione binaria, si sono settati i parametri d'appendimento in tal modo:
• Sigmoide, come funzione d'attivazione nello strato di output.
• Binary-crossentropy, come funzione di perdita.
• Accuracy, come funzione metrica.
• Adam, come funzione di ottimizzazione.

Una metrica è una funzione utilizzata per giudicare le prestazioni del modello. Una funzione metrica è simile a una funzione di perdita, tranne per il fatto che i risultati della valutazione di una metrica non vengono utilizzati durante l'addestramento del modello. È possibile utilizzare una qualsiasi delle funzioni di perdita come funzione metrica.

5.10.1 Risultati ottenuti

Prediction time	**Onset**	**12ore**	**24ore**	**48ore**
AUROC	0.89	0.84	0.79	0.75
Sensitivity	0.80	0.80	0.80	0.80
Specificity	0.91	0.72	0.58	0.50
Accuracy	0.89	0.73	0.61	0.53
DOR	40.65	10.41	5.66	3.97
LR+	8.93	2.88	1.93	1.59
LR-	0.21	0.27	0.34	0.40

Tabella 5.6. Risulati LSTM

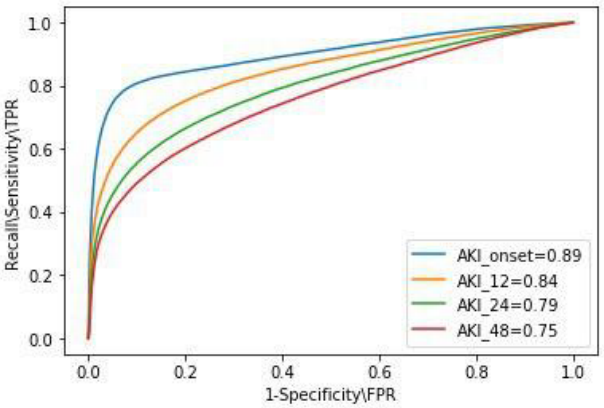


Figura 5.32. ROC

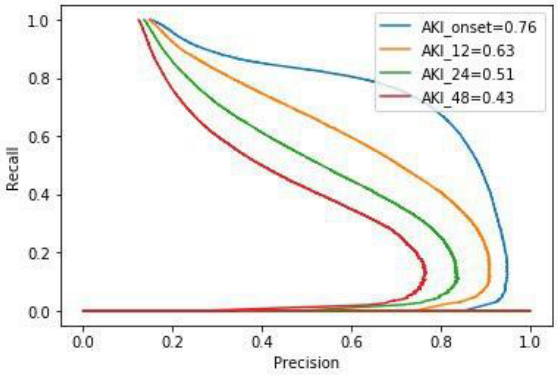


Figura 5.33. PRC

Capitolo 6

Conclusioni

Dopo aver preparato il dataset, spiegato in dettaglio nel Capitolo 3, si è proceduto alla vera e propria fase di classificazione esposta nel Capitolo 5.
Questo capitolo ha lo scopo di illustrare i risultati ottenuti nelle applicazioni degli algoritmi di classificazione, presentare un ulteriore convalida (10-fold validation), ed infine mostrare l'applicazione di un algoritmo su un'altra definizione di AKI definita come standard di riferimento e nota come National Health Service (NHS) England AKI Algorithm.

I vari algoritmi sono già implementati in Python, nella libreria scikit-learn. Si è quindi preparato il dataset nel seguente modo: un vettore di misurazioni di diuresi/peso e quattro class label binarie indicanti la presenza o meno di AKI sul momento e nelle 12/24/48 ore successive.
A questo punto una funzione (train_test_split) divide in maniera random la parte di dati da usare nel training (circa l'80%) e la parte di dati da usare nel test (il restante 20%). Tale convalida è meglio nota col nome "hold out".
E' importante che le etichette di classe presenti nel test set rispecchino in giusta proporzione quelle che caratterizzano il training set, ciò è indispensabile per essere certi di poter testare una certa popolazione in cui sia presente l'insufficienza renale, motivo principale dato che il dataset descritto è fortemente sbilanciato.
Alla fine di ogni run ogni classificatore è caratterizzato dai seguenti parametri :

- Auroc
- Sensitivity
- Specificity

- Accuracy
- DOR
- LR+
- LR−

Si è testata la capacità dei vari algoritmi di rilevare l'AKI all'insorgenza e di prevedere l'AKI 12, 24 e 48 ore prima dell'inizio.
Nelle rispettive Tabelle 6.1, 6.2, 6.3, 6.4 è possibile notare le performance degli algoritmi raggruppati per i punti temporali di interesse.

Nella Tabella 6.1 la maggioranza degli algoritmi presentano ottime performance, raggiungendo un AUROC di 0.90 per il rilevamento dell'AKI al momento dell'insorgenza. Non è di notevole importanza tuttavia testare un algoritmo su una class label riferita alla presenza di AKI o meno sul momento ai fini di una diagnosi precoce, in quanto quest'ultima farà fede sulla sola definizione di AKI.

	RegressioneLog	**KNN**	**RandomForest**	**XGBoost**	**MLP**	**LSTM**
AUROC	0.89	0.87	0.89	0.90	0.90	0.89
Sensitivity	0.80	0.78	0.80	0.80	0.80	0.80
Specificity	0.91	0.89	0.88	0.91	0.92	0.91
Accuracy	0.90	0.91	0.87	0.89	0.90	0.89
DOR	42.97	47.05	31.84	43.22	45.11	40.65
LR+	9.39	7.30	7.16	9.44	9.82	8.93
LR-	0.22	0.23	0.22	0.21	0.22	0.21

Tabella 6.1. Risulati relativi al momento dell'insorgenza di AKI

Per la previsione 12 ore prima dell'inizio (come presentato in Tabella 6.2), gli algoritmi migliori raggiungono un AUROC di 0.84. Si ricorda che un test è moderatamente accurato se presenta una AUROC compresa fra 0.70 e 0.90.
In termini di accuracy invece, KNN risulta il più performante presentando 0.81 tuttavia piazzandosi all'ultimo posto in termini di specificity, facendo emergere la sua scarsa accuratezza nel discriminare i veri negativi.

	RegressioneLog	**KNN**	**RandomForest**	**XGBoost**	**MLP**	**LSTM**
AUROC	0.84	0.80	0.82	0.84	0.84	0.84
Sensitivity	0.80	0.80	0.80	0.80	0.80	0.80
Specificity	0.71	0.57	0.66	0.71	0.73	0.72
Accuracy	0.72	0.81	0.68	0.73	0.74	0.73
DOR	9.94	10.40	7.82	10.23	10.72	10.41
LR+	2.79	1.92	2.36	2.84	2.94	2.88
LR-	0.28	0.33	0.30	0.27	0.27	0.27

Tabella 6.2. Risulati di previsione dell'AKI a 12 ore

	RegressioneLog	**KNN**	**RandomForest**	**XGBoost**	**MLP**	**LSTM**
AUROC	0.80	0.74	0.77	0.79	0.79	0.79
Sensitivity	0.80	0.76	0.78	0.80	0.80	0.80
Specificity	0.59	0.53	0.54	0.58	0.58	0.58
Accuracy	0.61	0.79	0.57	0.61	0.61	0.61
DOR	5.70	6.35	4.39	5.59	5.56	5.66
LR+	1.94	1.62	1.71	1.91	1.91	1.93
LR-	0.34	0.44	0.39	0.34	0.34	0.34

Tabella 6.3. Risulati di previsione dell'AKI a 24 ore

Per le previsioni a 24 ore (Tabella 6.3), la Regressione logistica raggiunge un valore di AUROC pari a 0.80, seguito da XGBoost, MLP e LSTM con 0.79.

Per le previsioni a 48 ore (Tabella 6.4), Regressione logistica, XGBoost ed MLP raggiungono valori di AUROC di 0.76. Quest'ultimi tuttavia presentano un accuracy di 0.53 circa, a differenza di KNN che raggiunge invece valori di accuracy di 0.76 e un valore di DOR pari a 4.55.

	RegressioneLog	KNN	RandomForest	XGBoost	MLP	LSTM
AUROC	0.76	0.70	0.73	0.76	0.76	0.75
Sensitivity	0.80	0.74	0.79	0.80	0.80	0.80
Specificity	0.50	0.48	0.45	0.50	0.50	0.50
Accuracy	0.53	0.76	0.49	0.53	0.54	0.53
DOR	3.91	4.55	3.20	4.04	4.04	3.97
LR+	1.58	1.45	1.45	1.60	1.61	1.59
LR-	0.40	0.51	0.45	0.39	0.40	0.40

Tabella 6.4. Risulati di previsione dell'AKI a 48 ore

LR- è un buon indicatore nell'escludere che la malattia sia presente ed è buona norma che sia inferiore ad 1 dal momento che è meno probabile che in soggetti malati il test abbia esito negativo rispetto a soggetti sani. In ogni finestra temporale, qualunque algoritmo presenta un valore di LR- inferiore a 1.
In tutti gli algoritmi considerati i valori di DOR sono tutti largamente superiori ad 1, e ciò è di notevole importanza dal momento che il test riesce a discriminare i pazienti con presenza di disturbo da quelli senza.
In conclusione, MLP in termini di AUROC, e anche in generale degli altri parametri, ha dimostrato un elevata AUROC per rilevare e prevedere l'AKI in base alla definizione KDIGO in tutti i punti temporali testati.

Poiché questo lavoro di tesi rappresenta uno studio retrospettivo, non si possono trarre forti conclusioni sull'impatto e sulle prestazioni degli algoritmi considerati in un contesto clinico.
Nonostante ciò il ruolo dell'apprendimento automatico in un contesto del genere può offrire importanti capacità diagnostiche consentendo quindi un intervento precoce da parte dei medici.

6.1 K-fold Cross Validation

Perché bisogna convalidare un modello? Una volta terminato l'addestramento di un determinato modello, non si può supporre che funzionerà bene su dati che non ha mai visto prima quindi non si può essere certi che il modello avrà l'accuratezza

desiderata nell'ambiente di produzione. Si vorrebbe una sorta di garanzia riguardante l'affidabilità delle previsioni generate dal modello.
La validazione, quindi ricopre un ruolo indispensabile in quanto rappresenta il processo per decidere se i risultati numerici che quantificano le relazioni ipotizzate tra variabili sono accettabili.
Fra le varie tecniche di validazione spiccano:

1. Train-Test split
   In questo caso i dati vengono divisi casualmente in training set e test set. Il modello si allena sul set di addestramento, e il test set viene usato come convalida. Idealmente i dati vengono divisi in 70:30 o 80:20.

2. K-fold cross validation

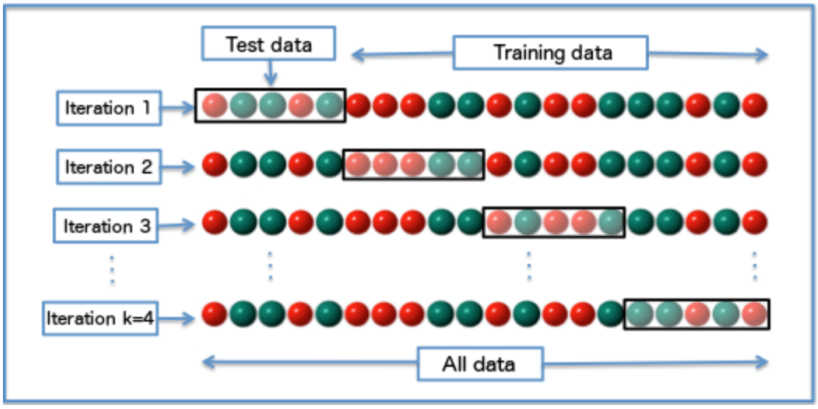


Figura 6.1. K-fold cross validation

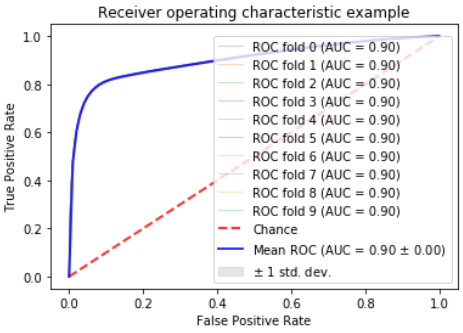
In tale validazione i record sono suddivisi in gruppi, in maniera casuale, e ciascuno di essi viene utilizzato k-1 volte per la creazione del modello ed una volta per la validazione. Le prestazioni finali quindi sono date dalla media delle k singole valutazioni che si ottengono. In questo modo quindi si garantisce che ogni osservazione presente nell'insieme di dati originale abbia la possibilità di essere presente sia nel training che nel test set. Quest'ultimo si classifica come uno dei migliori approcci se si dispone di dati di input limitati. Di seguito i passi salienti:

- Dividere tutti i dati in modo casuale in k-folds (il valore di k non dovrebbe essere troppo piccolo o troppo alto, idealmente si sceglie da 5

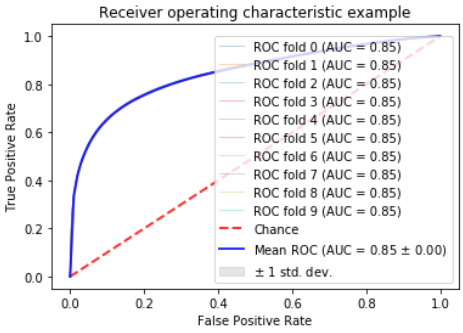
a 10 a seconda della dimensione dei dati). Un valore di K alto porta a un modello meno distorto (ma una grande varianza potrebbe portare a overfit), mentre un valore basso di K è simile all'approccio train-test split.

- Addestrare il modello usando K-1 folds e convalidare il modello usando il K^{th} fold rimanente. Annotare i punteggi/errori.
- Ripetere questo processo fino a quando ogni fold funge da set di test. La metrica delle prestazioni per il modello sarà la media dei punteggi registrati.

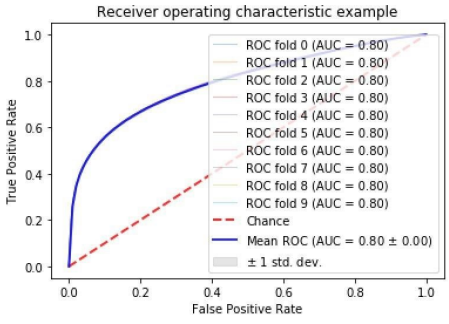
In questo studio si è scelto di validare ulteriormente l'algoritmo MLP in tutti gli instanti temporali tramite la convalida 10-fold cross validation. Nella Figura 6.2 vengono mostrate le ROC e le rispettive AUROC per ogni fold.



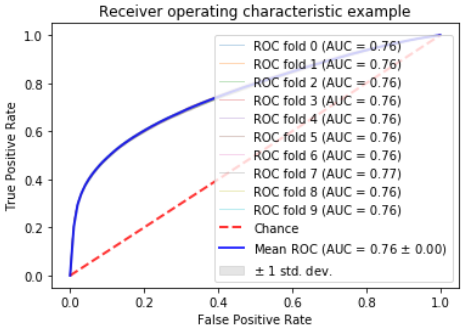
ROC AKI onset



ROC AKI 12ore



ROC AKI 24ore



ROC AKI 48ore

Figura 6.2. Risultati 10-fold cross validation per MLP

Si nota che le AUC in ogni fold assumono lo stesso valore, deducendo quindi che il modello non è legato a una determinata parte di training set ma riesce bene

a generalizzare. I risultati ottenuti tramite 10-fold validation sono un'ulteriore conferma della non casualità delle performance ottenute tramite la convalida train-test split.

6.2 National Health Service England AKI Algorithm

Il National Health Service (NHS) England AKI Algorithm è stato implementato come *gold standard*. Tale algoritmo è basato fondamentalmente sulle linee guida Kidney Disease concentrandosi tuttavia esclusivamente sui cambiamenti nei livelli di SCr per determinare la presenza e la stadiazione di AKI anziché sulla produzione di urina.
Si è ritenuto utile classificare i pazienti presenti nel dataset anche secondo questo standard in modo tale da avere un confronto con i criteri KDIGO.
La Figura 6.3 mostra un flow chart riguardante la metodologia di scelta di valore basale di creatina e i relativi criteri di stadiazione dell'AKI.
Come mostrato in Figura 6.3 il valore basale di creatinina può essere calcolato facendo riferimento al valore minimo di creatinina negli ultimi sette giorni, oppure estrarre il valore mediano tra gli ultimi 8 fino a 365 giorni.

Tuttavia in questo studio, sempre sotto consiglio di un esperto in nefrologia, si è adottata la scelta di considerare come valore basale la prima misurazione di creatinina all'ammissione in degenza ospedaliera del paziente.
Si è calcolato lo SCr ratio come : (creatina corrente)/(valore basale di creatina)
E' importante sottolineare che le scelte implementative riguardanti la creazione del dataset, spiegata in dettaglio nel Capitolo 3, rimangono invariate quindi in input si avrà sempre un vettore di sei misurazioni di diuresi/peso e la class label farà fede ai criteri descritti dal gold standard. Il dataset ottenuto risulta esser ancor più sbilanciato rispetto a quello riguardante le linee guida Kdigo.

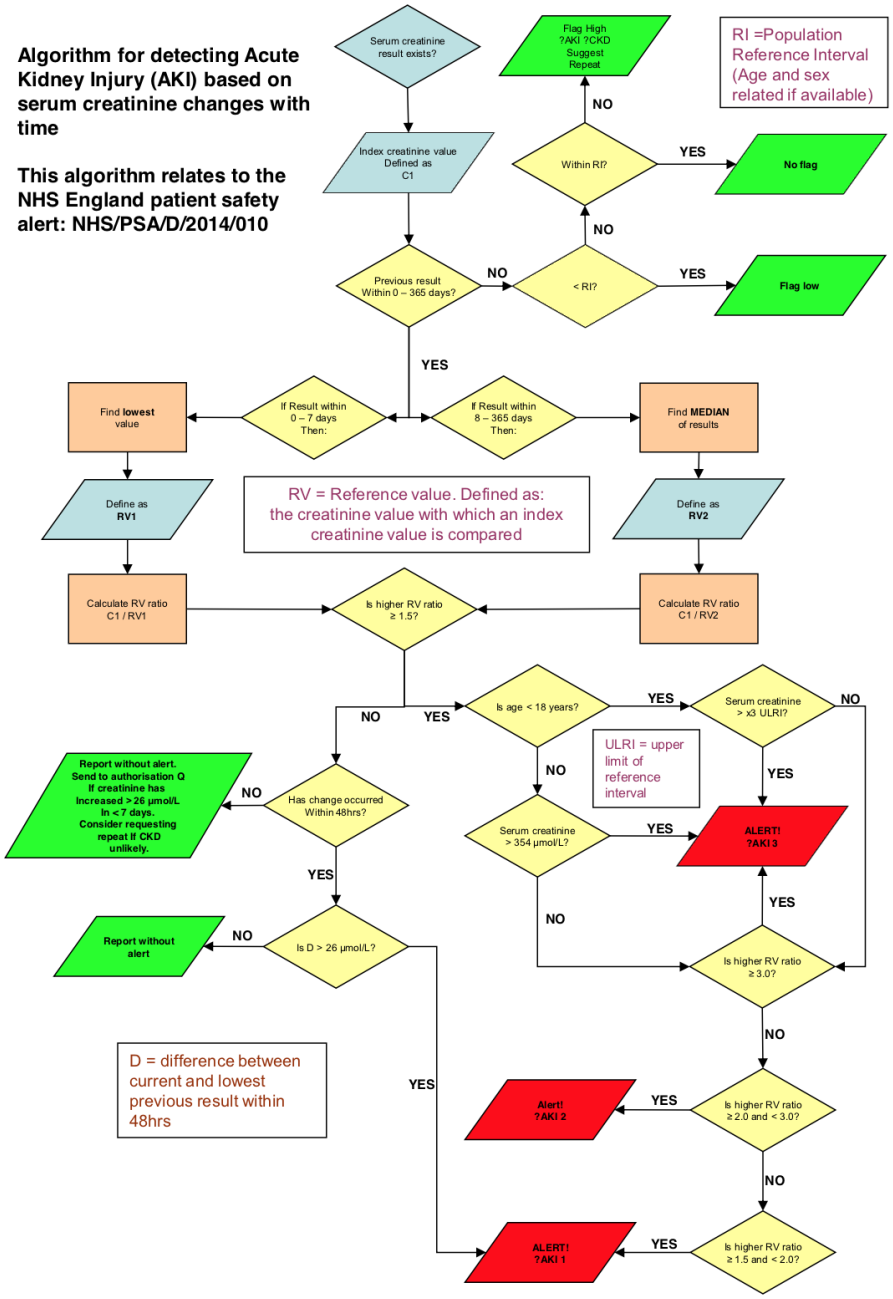


Figura 6.3. Gold standard

Sono stati testati gli algoritmi di XGBoost e MLP con la nuova class label definita e i relativi risultati ottenuti sono mostrati nelle Tabelle 6.5 e 6.6 .

Prediction time	Onset	12ore	24ore	48ore
AUROC	0.70	0.71	0.71	0.70
Sensitivity	0.80	0.80	0.80	0.80
Specificity	0.42	0.43	0.42	0.42
Accuracy	0.45	0.45	0.44	0.44
DOR	2.97	3.04	2.94	2.91
LR +	1.39	1.41	1.39	1.38
LR-	0.47	0.46	0.47	0.47

Tabella 6.5. Risulati XGBoost con gold label

Prediction time	Onset	12ore	24ore	48ore
AUROC	0.69	0.69	0.70	0.69
Sensitivity	0.80	0.80	0.80	0.80
Specificity	0.40	0.40	0.40	0.39
Accuracy	0.40	0.42	0.43	0.41
DOR	2.65	2.69	2.72	2.52
LR +	1.33	1.34	1.34	1.30
LR-	0.50	0.50	0.49	0.52

Tabella 6.6. Risulati MLP con gold label

Da come si può notare i risultati ottenuti in entrambi gli algoritmi presentano valori di AUROC quasi identici in tutti i punti temporali considerati.
A differenza dei risultati ottenuti mediante i criteri Kdigo, questi non risentono l'avanzamento del periodo temporale di previsione. Ciò è dovuto al fatto che le etichette assegnate al momento dell'insorgenza di AKI non subiscono eccessive variazioni nelle successive 12/24/48 ore. Quindi è come avere le stesse label assegnate in tutti i punti temporali. Questo viene dimostrato a fronte di un indagine effettuata sulle etichette per verificare i cambiamenti di stato (sano/malato e malato/sano) che si hanno durante la degenza dei pazienti.
Nella Figura 6.4 si mostra un istogramma rappresentante i conteggi a fronte dei cambiamenti di stato e non avvenuti nelle 48 ore.

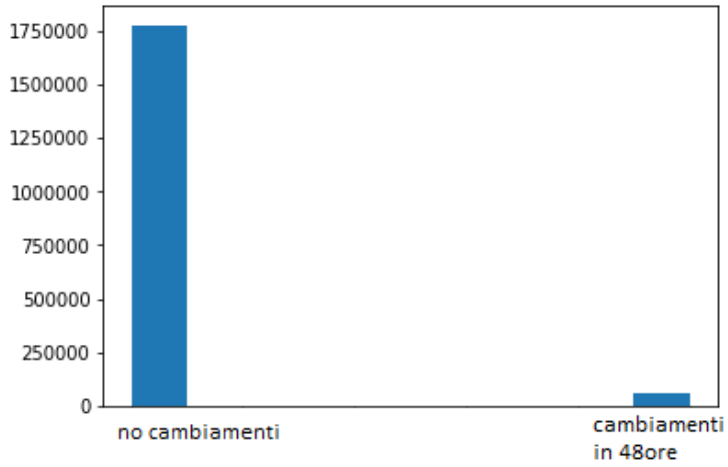


Figura 6.4. Conteggio cambiamenti di stato in 48 ore

6.3 Sviluppi futuri

Poichè in questo studio si sono considerati nella grande maggioranza pazienti statunitensi di età superiore ai 18 anni presenti nel Beth Israel Deaconess Medical Center di Boston, i risultati potrebbero non essere generalizzabili ad altre popolazioni.
Per tal motivo sarebbe interessante considerare dati relativi ad altre popolazioni, ovvero etnicamente diversificate, in modo da poter generalizzare il lavoro svolto.
Si ricorda che i criteri KDIGO e GOLD usano rispettivamente creatinina/diuresi e la sola creatinina per classificare i pazienti: per tal motivo si suggerisce di provare a prendere in considerazione nuovi parametri fisiologici (frequenza cardiaca, pressione sanguigna o temperatura corporea) o sfruttare l'influenza del bilancio idrico, l'uso di diuretici e l'applicazione di differenti stime del peso corporeo (reale, ideale, massa magra) come input agli algoritmi di machine learning.
Inoltre si potrebbero aggiungere altre informazioni del paziente riguardanti sesso, età, razza ed una feature indicante la presenza di un'altro disturbo (come ad esempio la sepsi, considerata la condizione più frequente associata con AKI nelle unità di terapia intensiva e che può determinare alterazioni della funzione renale senza alcun cambiamento dei parametri urinari) al fine di aumentare la capacità predittiva dei modelli.
Si potrebbero discriminare i pazienti, aggiungendo in ogni record una nuova feature riguardante lo stato di salute del paziente all'ammissione e alla fine della degenza:

sano/sano, sano/malato, malato/sano, malato/malato.
In tal modo ci si potrebbe concentrare su un'analisi riguardante pazienti che cambiano stato nel corso della loro degenza soprattutto il cambiamento di stato sano/malato ritenuto di maggiore importanza ai fini di una diagnosi precoce.

Bibliografia

[1] *Linee guida alla prevenzione, diagnosi e terapia delle sindromi di danno renale acuto: versione italiana delle KDIGO, integrata con le nuove evidenze e i commentari internazionali. SEZIONE 1.* URL: http://www.siaarti.it/SiteAssets/Ricerca/Linee-Guida-SIN-SIAARTI/Sezione%201.pdf.

[2] "Danno renale acuto, un nuovo test per la diagnosi precoce". In: *Sanità24 - Il Sole 24 Ore* (apr. 2019). URL: https://www.medicinadimed.unipd.it/sites/medicinadimed.unipd.it/files/Danno%20renale%20acuto%2C%20un%20nuovo%20test%20per%20la%20diagnosi%20precoce%20_%20Sanit%C3%A024%20-%20Il%20Sole%2024%20Ore.pdf.

[3] Hilde R. H. de Geus, Michiel G. Betjes e Jan Bakker. "Biomarkers for the prediction of acute kidney injury: a narrative review on current status and future challenges". In: *Clinical Kidney Journal* 5.2 (apr. 2012), pp. 102–108. ISSN: 2048-8505. DOI: 10.1093/ckj/sfs008. eprint: https://academic.oup.com/ckj/article-pdf/5/2/102/979651/sfs008.pdf. URL: https://doi.org/10.1093/ckj/sfs008.

[4] CV Thakar, A Christianson e R Freyberg. "Incidence and outcomes of acute kidney injury in intensive care units: a Veterans Administration study". In: *Crit Care Med* 37.9 (set. 2009), pp. 2552–2558. DOI: 10.1097/CCM.0b013e3181a5906f.

[5] Claudio Ronco et al. "Improving outcomes from acute kidney injury (AKI): Report on an initiative". In: *The International journal of artificial organs* 30 (giu. 2007), pp. 373–6. DOI: 10.4103/0971-4065.35011.

[6] José António Lopes e Sofia Jorge. "The RIFLE and AKIN classifications for acute kidney injury: a critical and comprehensive review". In: *Clinical Kidney Journal* 6.1 (gen. 2012), pp. 8–14. ISSN: 2048-8505. DOI: 10.1093/ckj/

sfs160. eprint: https://academic.oup.com/ckj/article-pdf/6/1/8/1270102/sfs160.pdf. URL: https://doi.org/10.1093/ckj/sfs160.

[7] *Linee guida alla prevenzione, diagnosi e terapia delle sindromi di danno renale acuto: versione italiana delle KDIGO, integrata con le nuove evidenze e i commentari internazionali. SEZIONE 2.* URL: http://www.siaarti.it/SiteAssets/Ricerca/Linee-Guida-SIN-SIAARTI/Sezione%202.pdf.

[8] G Eknoyan, N Lameire e R Barsoum. "The burden of kidney disease: Improving global outcomes". In: *Kidney International* 66.4 (ott. 2004), pp. 1310–4. DOI: 10.1111/j.1523-1755.2004.00894.x. URL: https://doi.org/10.1111/j.1523-1755.2004.00894.x.

[9] SS Soni, R Pophale e C Ronco. "New biomarkers for acute renal injury". In: *Clin Chem Lab Med* 49 (lug. 2011). DOI: https://doi.org/10.1515/CCLM.2011.664.

[10] P Milani, MS Graziani e G. Merlini. "Marcatori di danno renale acuto". In: *Biochim Clin* 34 (2010), pp. 585–90. DOI: https://doi.org/10.1515/CCLM.2011.664.

[11] A Clerico, C Galli e A et al. Fortunato. "Neutrophil gelatinase-associated lipocalin (NGAL) as biomarker of acute kidney injury: a review of the laboratory characteristics and clinical evidences". In: *Clin Chem Lab Med* 50 (2012), pp. 1505–17. DOI: https://doi.org/10.1515/cclm-2011-0814.

[12] R Bellomo, C Ronco e JA et al. Kellum. "Acute renal failure - definition, outcome measures, animal models, fluid therapy and information technology needs: the Second International Consensus Conference of the Acute Dialysis Quality Initiative ADQJ Group". In: *Crit Care* 8 (2004), R204–12. DOI: 10.1186/cc2872.

[13] C Ronco, P Mc Cullough e SD Anker. "Cardio-renal syndromes: report from the consensus conference of the Acute Dialysis Quality Initiative". In: *Eur Heart J* 31 (2010), pp. 703–11. DOI: 10.1093/eurheartj/ehp507.

[14] CL Weber, M Bennett e L et al. Er. "Urinary NGAL levels before and after coronary angiography: a complex story". In: *Nephrol Dial Transplant* 26 (2011), pp. 3207–11. DOI: 10.1093/ndt/gfr033.

[15] SS Khamis, AG Dala e HA et al. Ahmad. "Study of urine neutrophil gelatinase associated lipocalin (NGAL) in post percutaneous coronary intervention contrast induced acute kidney injury". In: *J Am Sci* 8 (2012), pp. 722–30.

[16] PA McCullogh, FJ Williams e DN et al Stivers. "Neutrophil gelatinase-associtaed lipocalin: a novel marker of contrast nephropathy risk". In: *Am J Nephrol* 35 (2012), pp. 509–14.

[17] M Haase, R Bellomo e P et al Devarajan. "NGAL Meta- analysis Investigator Group. Accuracy of neutrophil gelatinase-associated lipocalin (NGAL) in diagnosis and prognosis in acute kidney injury: a systematic review and metaanalysis". In: *Am J Kidney Dis* 54 (2009), pp. 1012–24.

[18] G Lippi e M Plebani. "Neutrophil gelatinase-associated lipocalin (NGAL): the laboratory perspective". In: *Clin Chem Lab Med* 50 (2012), pp. 1483–7. DOI: https://doi.org/10.1515/cclm-2012-0344.

[19] Y Tomonaga, T Szucs e P et al. Ambűhl. "Insights on urinary NGAL obtained in a primary care setting". In: *Clin Chim Acta* 413 (2012), pp. 733–9. DOI: 10.1016/j.cca.2012.01.001.

[20] URL: https://www.nephrocheck.com/us/.

[21] JL Vincent, R Moreno e J Takala. "The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure. On behalf of the working group on sepsis-related problems of the European Society of Intensive Care Medicine". In: *Intensive Care Med* 22 (1996), pp. 707–710. DOI: 10.1007/BF01709751.

[22] NM Selby, R Hill e RJ Fluck. "Standardizing the early identification of acute kidney injury: the NHS England national patient safety alert". In: *Nephron* 131 (2015), pp. 113–117. DOI: 10.1159/000439146.

[23] N. Tomašev, X. Glorot e J.W. Rae. "A clinically applicable approach to continuous prediction of future acute kidney injury". In: *Nature* 572 (2019), pp. 116–119. DOI: 10.1038/s41586-019-1390-1.

[24] A. Johnson, T. Pollard e L. et al Shen. "MIMIC-III, a freely accessible critical care database". In: *Sci Data* (). DOI: https://doi.org/10.1038/sdata.2016.35.

[25] JA Swets. "Measuring the accuracy of diagnostic systems". In: *Science* 240 (1998), pp. 1285–93. DOI: 10.1126/science.3287615.

[26] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[27] URL: `https://towardsdatascience.com/understanding-random-forest-58381e0602d2`.

Ringraziamenti

Mi è doveroso dedicare uno spazio speciale del mio elaborato alle persone che hanno contribuito, con il loro instancabile supporto, alla realizzazione dello stesso.
Innanzitutto, ringrazio la Professoressa Valentina Alice Cauda per aver accettato l'incarico di relatrice per la mia tesi, il correlatore Eros Gian Alessandro Pasero, e i dottorandi Andrea, Vincenzo e Nancy sempre pronti ad aiutarmi. Grazie per le conoscenze trasmesse durante lo svolgimento di questo lavoro di tesi.
Ringrazio infinitamente i miei genitori per avermi fatto raggiungere questo strepitoso traguardo. Grazie per avere sempre creduto in me.
Ringrazio Angelo, persona importantissima con cui ho condiviso tutto in questi anni della mia vita. Solo una pandemia poteva separarci, ma sono sicuro che i nostri destini si incrocieranno di nuovo. Grazie di cuore amico mio.
Ringrazio Torino, tutte le persone che ho avuto modo di conoscere, soprattutto Marco, Davide, Pietro, Rocco e i colleghi Francesco,Salvo, Giovanni,Eros,Marco. Grazie amici, perchè tutti insieme siete stati come una famiglia.