



UNIVERSITÀ
DI SIENA
1240

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE E SCIENZE
MATEMATICHE

Corso di laurea in
Ingegneria Informatica e dell'Informazione

Predizione degli Effetti Avversi dei Farmaci Tramite Reti Neurali Artificiali

Relatore:
Prof. Monica Bianchini

Candidato:
Carlo Merola

Anno Accademico 2021-2022

Dedica

Dedico questo studio a Wally - Luigi -, che al momento della scrittura, si trova in condizioni di salute gravi. E alla sua famiglia, Emma, Edoardo ed Eleonora. Hanno sempre mostrato tanta forza, propria di poche persone.

Indice

Dedica	1
1 Introduzione	4
1.1 Struttura del documento	5
2 Machine Learning e Classificazione	6
2.1 Machine Learning	6
2.2 Il Multilayer Perceptron	7
2.3 Apprendimento Supervisionato e Non Supervisionato	7
2.4 Classificazione	9
3 Metodologia	11
3.1 Costruzione del Dataset	11
3.1.1 Dataset: Vettore delle Feature	11
3.1.2 Dataset: Vettore Target	13
3.1.3 Partizione del Dataset	16
3.2 Costruzione della Rete Neurale Artificiale e Definizione degli Iperpa- rametri	16
3.2.1 Rete Neurale Artificiale: gli Iperparametri	18
3.2.2 Rete Neurale Artificiale: Implementazione	19
3.2.3 Il Problema delle Classi Sbilanciate	20

3.2.4	Libreria di Compilazione	22
4	Risultati	23
4.1	Metodi di Valutazione	23
4.2	Prestazioni del Modello Predittivo	27
5	Conclusioni	31
	Appendice	32
	Riferimenti	40

Capitolo 1

Introduzione

L'oggetto di questa tesi riguarda lo sviluppo di un sistema per prevedere gli effetti collaterali dei farmaci — a partire dalle caratteristiche strutturali della molecola —, un problema particolarmente importante nel drug design.

Le molecole possono infatti essere esaminate con metodi computazionali prima di essere sottoposte a sperimentazione con studi clinici, onde evitare inutili costi aggiuntivi e rischi per la salute dei partecipanti.

Per prelevare i dati atti allo studio, mi sono avvalso dei database pubblici SIDER e PubChem. Il primo contiene informazioni sui farmaci in commercio e sulle relative reazioni avverse registrate. Le informazioni sono estratte da documenti pubblici e foglietti illustrativi [1]. Pubchem, invece, è un database chimico liberamente accessibile, messo a disposizione dal National Institutes of Health (NIH). Pubchem è diventato una risorsa fondamentale di informazioni chimiche per scienziati, studenti e per il pubblico in generale [2].

Il programma sviluppato per la predizione degli effetti avversi dei farmaci utilizza tecniche di Machine Learning, e, in particolare, sfrutta le reti neurali artificiali.

1.1 Struttura del documento

La tesi è organizzata come descritto di seguito.

- Nel Capitolo 2, viene proposta una breve introduzione al Machine Learning, alle Reti Neurali Artificiali e alla tipologia di classificazione utilizzata.
- Nel Capitolo 3 viene spiegato il metodo proposto, partendo dall'elaborazione dei dati, fino alla costruzione del modello di Rete Neurale Artificiale e alla selezione degli iperparametri della rete.
- I risultati ottenuti vengono presentati nel Capitolo 4, insieme ad un'analisi sul metodo valutativo.
- Infine, il Capitolo 5 contiene le conclusioni sul lavoro svolto e ne delinea i possibili sviluppi futuri.

In Appendice, è riportato il codice nel linguaggio di programmazione Python, con la relativa spiegazione delle funzioni più importanti.

Capitolo 2

Machine Learning e Classificazione

Questo capitolo introduce il concetto di Machine Learning, seguito da un approfondimento sulle Reti Neurali Artificiali e i diversi framework di classificazione dei dati.

2.1 Machine Learning

Il Machine Learning (ML) è una branca dell'Intelligenza Artificiale (AI) che si occupa di creare sistemi che apprendono o migliorano le performance in base ai dati che utilizzano [3]. Le tecniche di Machine Learning, quindi, consentono al modello di operare senza che la sua logica venga esplicitamente programmata con regole predefinite. Questo paradigma differisce dall'algoritmo nella sua forma tradizionale, che prende dei dati in input, e restituisce un output deterministico basato su una funzione scritta in forma esplicita [4].

L'approccio algoritmico convenzionale è una forma di *Hard Computing*. Il principale svantaggio dell'hard computing è che, talvolta, è inadatto a risolvere i problemi del mondo reale, il cui comportamento è impreciso e presenta elevata variabilità nei dati. Di contro, l'approccio *Soft Computing* del Machine Learning è caratterizzato dalla

tolleranza per l'imprecisione, l'incertezza e la verità parziale per ottenere trattabilità del problema e robustezza del modello (dalla definizione di Lofti Zadeh) [5].

2.2 Il Multilayer Perceptron

Una Rete Neurale Artificiale Multilayer Perceptron (MLP) impiega la metafora neurobiologica a livello della singola unità elementare, il neurone artificiale. Queste unità o nodi sono densamente interconnesse tra loro. Ogni connessione, come le sinapsi in un cervello biologico, può trasmettere un segnale ad altri neuroni. Un neurone artificiale riceve il segnale (un numero reale) e ne elabora l'output attraverso una funzione non lineare della somma dei suoi input. Le connessioni fra neuroni hanno tipicamente un peso che si adatta man mano che l'apprendimento procede, e conseguentemente, aumenta o diminuisce la forza del segnale corrispondente che viene prodotto in output [6].

In particolare, negli MLP i neuroni sono aggregati in strati. I segnali viaggiano dal primo livello (il livello di input) all'ultimo livello (il livello di output). In mezzo si trovano uno o più strati di neuroni, chiamati strati "hidden" (stati nascosti), il cui nome deriva dal fatto che l'attività delle unità nascoste non è visibile dall'esterno del modello. Una semplice raffigurazione di una rete con singolo strato nascosto, è mostrata in Fig. 2.1.

2.3 Apprendimento Supervisionato e Non Supervisionato

Il Machine Learning non supervisionato utilizza un approccio in cui il modello impara a identificare processi e schemi complessi senza specificare nessuna informazione sulla loro corretta interpretazione. Implica un training basato su dati privi di eti-

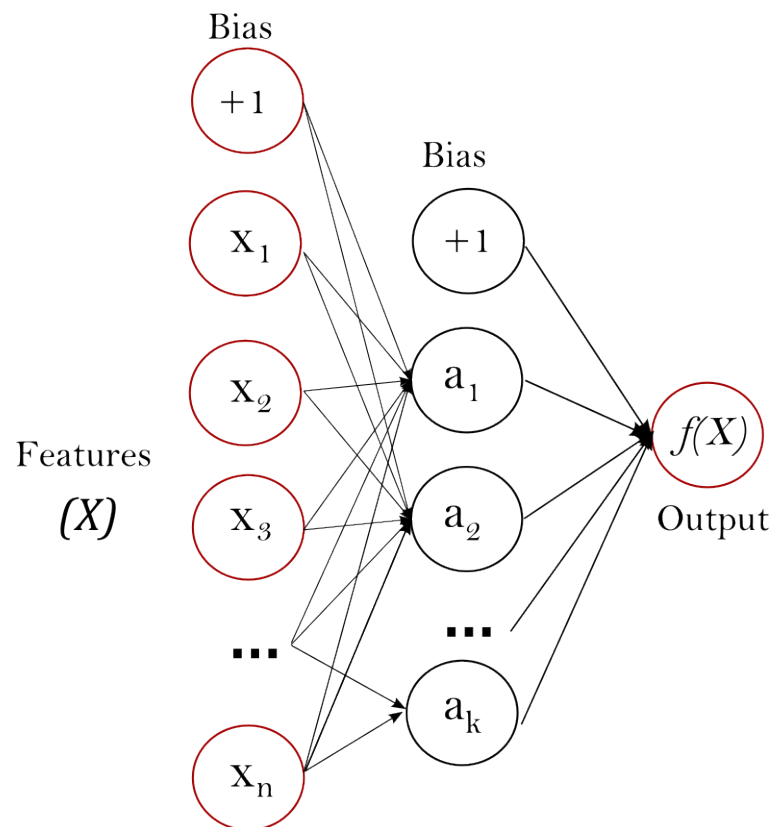


Figura 2.1: Struttura di una semplice rete neurale, composta da uno strato di input, un solo strato hidden, e uno strato di output. La profondità della rete aumenta in conseguenza al numero di strati hidden presenti.

chette e per i quali non è stato definito un output specifico [3].

Viceversa, se l'apprendimento è supervisionato, per ogni esempio in ingresso è specificata la corretta rappresentazione o etichetta corrispondente in output, detto valore "target". L'errore commesso dalla rete — valutato come differenza fra il target e l'output calcolato —, viene propagato all'indietro, nel verso opposto alla propagazione del segnale di ingresso, per aggiornare i pesi e imparare una funzione sui dati. Il processo è chiamato *backpropagation* e fonda la sua idea di principio sul fatto che, se l'errore cambia in base ai pesi dei neuroni, allora può essere espresso come funzione dei pesi stessi. Di detta funzione verranno quindi calcolati i punti di minimo. Mediante apprendimento supervisionato, è possibile fare predizioni o classificazioni sui dati di input. Questo costituisce anche una delle differenze fondamentali rispetto alla soluzione algoritmica di problemi: una volta che il modello ha elaborato la funzione, identificando i pattern sui dati di ingresso, il modello deve avere capacità di generalizzare ad input che non sono stati visti precedentemente, in modo tale che le predizioni siano le più corrette possibili.

In questa tesi, il paradigma di apprendimento supervisionato è applicato al problema della predizione degli effetti collaterali dei farmaci.

2.4 Classificazione

Esistono due principali classi di problemi che si possono approcciare con tecniche di apprendimento supervisionato: classificazione e regressione.

Nella regressione l'obiettivo è quello di approssimare il comportamento di una funzione i cui valori sono noti su un campione di esempi. Invece, l'obiettivo della classificazione è prevedere un'etichetta categorica per un esempio di input. I dati in ingresso sono divisi in classi diverse e l'algoritmo viene addestrato per prevedere la

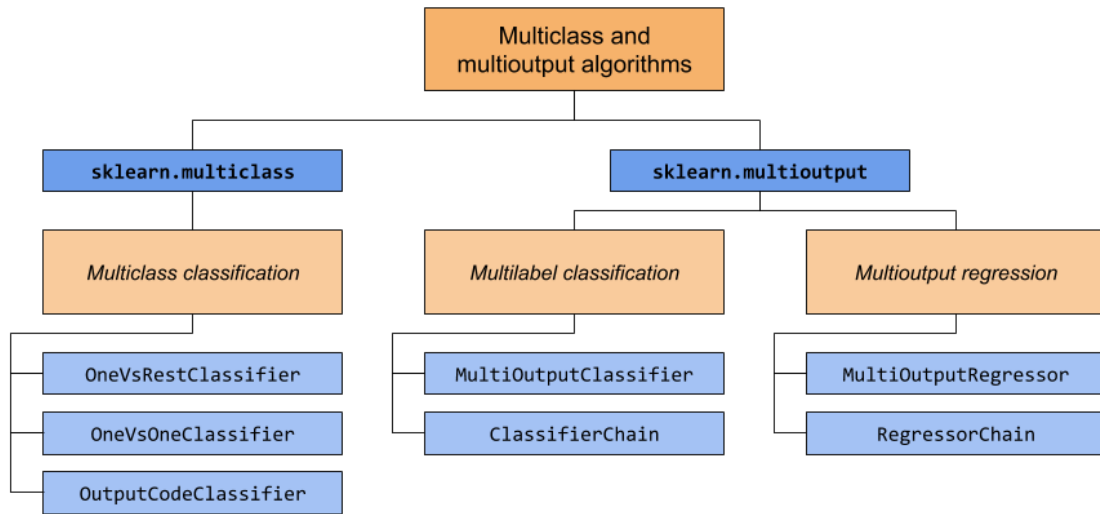


Figura 2.2: Categorie di algoritmi "multi-learning", compresi multiclasse, multioutput e multilabel, della libreria Scikit-Learn.

classe di appartenenza di un nuovo campione.

Tra le diverse tipologie di classificazione, il problema in esame si colloca nella categoria di classificazione binaria multi-label. La rete assegna, ad ogni campione, m etichette da n possibili classi, dove m può essere compreso tra 0 e n , il che corrisponde a predire proprietà di un campione che non sono mutuamente esclusive, ovvero sono indipendenti l'una dall'altra. Formalmente, un output binario viene assegnato a ciascuna classe, per ogni campione. Le classi positive sono indicate con 1 e le classi negative con 0 [7].

Il significato assunto dal vettore binario in uscita, riguarda l'assegnazione di etichette multiple, o classi positive, ai campioni in ingresso. In questo caso quindi, le etichette sono i possibili effetti collaterali che possono verificarsi con il dato farmaco.

Capitolo 3

Metodologia

In questo capitolo verrà esaminato il procedimento utilizzato per risolvere il problema della predizione degli effetti collaterali delle molecole candidate all'uso farmaceutico.

3.1 Costruzione del Dataset

Le Reti Neurali Artificiali usano *esempi* durante il loro processo di apprendimento. In questo caso i pattern utilizzati sono costituiti dai dati strutturali delle molecole e dagli effetti collaterali da esse prodotti.

3.1.1 Dataset: Vettore delle Feature

I vettori delle caratteristiche, dette **feature** (X), andranno in ingresso alla rete (vedi Figura 2.1). Le informazioni sulle caratteristiche molecolari dei farmaci sono state prelevate dal database PubChem, grazie al quale sono stati ricavati sette descrittori chimici, oltre alle stringhe SMILES (Simplified Molecular Input Line Entry System) di ciascuna molecola.

Le sette caratteristiche molecolari rappresentano: il peso molecolare (mw), l'area po-

lare superficiale (polararea), il coefficiente di ripartizione (xlogp), il numero di atomi pesanti (heavycnt), il numero di legami idrogeno donati (hbonddonor), il numero di legami idrogeno accettati (hbondacc) e il numero di legami con libera rotazione (rotbonds). Queste caratteristiche sono state normalizzate per assumere valori compresi tra $[0, 1]$, in modo da influenzare in stessa proporzione l'apprendimento.

La standardizzazione di un set di dati è un requisito comune per molti stimatori di Machine Learning. In genere questo viene fatto rimuovendo la media e ridimensionando la varianza in modo che assuma valore unitario [13]. Tuttavia, i valori anomali possono spesso influenzare la media/varianza campionaria in modo negativo. Per questo motivo ho posto il vettore delle caratteristiche sotto processo di standardizzazione, rimuovendo il valore mediano e ridimensionando i dati in base agli intervalli quantili del 25% e 75%, in modo da ridurre l'effetto negativo dei valori singolari molto lontani dalla media. Successivamente ogni caratteristica è stata ulteriormente tradotta in un valore che si collocasse all'interno dell'intervallo dato dalla formula:

$$x_{i-std} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

dove x_i è il valore numerico della caratteristica presa in esame per l' i -esimo campione, x_{min} è il valore minimo assunto dalla caratteristica e x_{max} ne è il valore massimo.

Le stringhe SMILES sono brevi stringhe ASCII che descrivono la struttura di una molecola. Da queste sono stati ottenuti i "fingerprint" descrittivi delle molecole, grazie alla libreria per l'analisi molecolare RDKit. I fingerprint molecolari sono vettori binari altamente discriminanti che descrivono le caratteristiche strutturali di una molecola. Sono spesso usati nella chimica computazionale e nella scoperta di farmaci per confrontare e identificare le molecole.

L'algoritmo di fingerprinting di RDKit implementa una funzione di hashing che

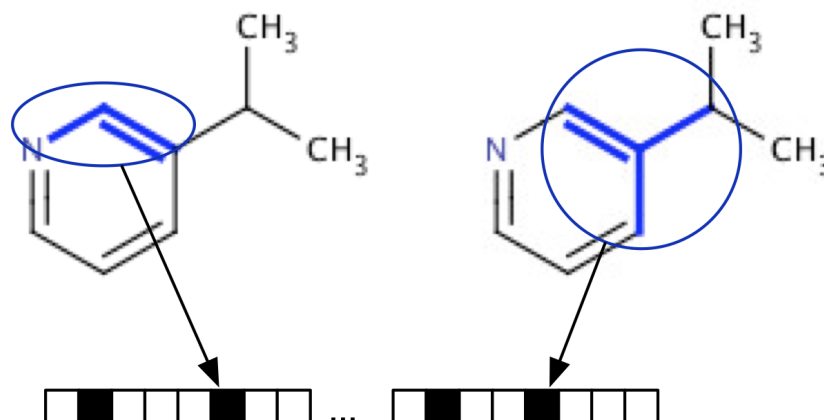


Figura 3.1: Ogni bit del vettore di fingerprint corrisponde a un frammento della molecola.

identifica tutti i sottografi nella molecola all'interno di un particolare intervallo di dimensioni. Dopodiché esegue l'hashing di ciascun sottografo per generare un bit ID grezzo. Modifica poi il bit grezzo per adattarlo alla lunghezza del vettore assegnata e quindi imposta il bit corrispondente (Figura 3.1). Lo schema predefinito per l'hashing dei sottografi consiste nell'hashing dei singoli legami in base a i tipi dei due atomi, l'angolo di legame in gradi e il tipo di legame [14].

3.1.2 Dataset: Vettore Target

Le informazioni sono state prelevate consultando in prima istanza il database SIDER, contenente le reazioni avverse dei medicinali attualmente in commercio. Queste informazioni costituiscono, dopo essere state ulteriormente elaborate, i **target** (**y**) utilizzati per l'apprendimento supervisionato.

In particolare, il database contiene in ogni riga l'identificativo del composto, sia per la rappresentazione chimica piana (flat id), sia tridimensionale (stereo id). Il primo tipo di identificativi CID (Compound ID number) inizia con la stringa CID1, il se-

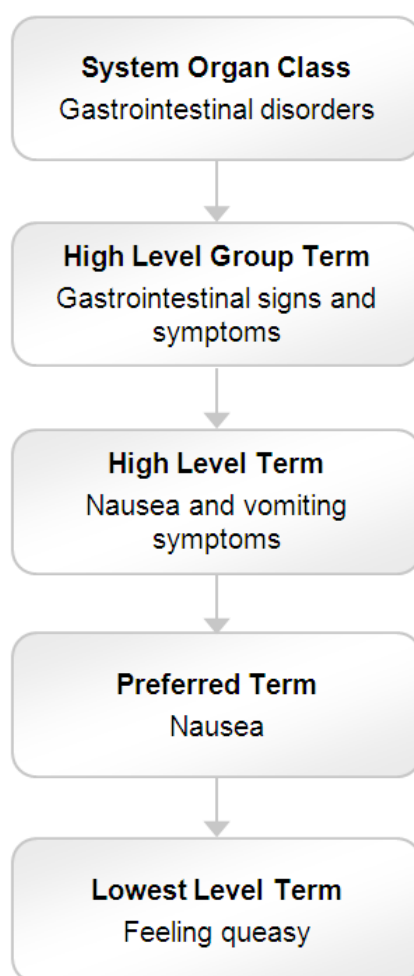


Figura 3.2: I livelli gerarchici di MedDRA [11].

condo con CID0. Inoltre, sono presenti l'UMLS Concept ID dell'effetto collaterale, il nome per esteso e la sua tipologia nella gerarchia MedDRA [8].

L'UMLS, o Unified Medical Language System, è un insieme di file e software che riunisce molti vocabolari e standard sanitari e biomedici per consentire l'interoperabilità tra i sistemi informatici, e ne distribuisce la terminologia chiave [10].

MedDra (Medical Dictionary for Regulatory Activities), invece, è una terminologia medica internazionale clinicamente validata, usata dalle autorità regolatorie e dalle industrie biofarmaceutiche. MedDRA presenta una struttura gerarchica organizzata su 5 livelli (vedi Figura 3.2). Nel database SIDER sono presenti sia le terminologie

degli effetti collaterali che si collocano nella categoria gerarchica più bassa, "LLT" ("Lowest Level Terms"), sia terminologie appartenenti al livello gerarchico appena superiore, "PT" ("Preferred Terms"). Ogni termine "LLT" è associato ad una singola voce "PT", che rappresenta a sua volta un singolo concetto medico. Viceversa un termine proprio della categoria "PT" può avere un numero illimitato di elementi "LLT" associati. Infatti, i termini di più basso livello includono sinonimi, varianti lessicali e varianti "colloquiali" dei termini "PT" [12].

Per questo motivo il primo passo nell'elaborazione dei dati è consistito nel filtrare e prelevare unicamente i termini degli effetti collaterali appartenenti alla categoria dei "Preferred Terms". In seconda istanza, si è utilizzata una funzione per filtrare gli effetti collaterali in base al numero di occorrenze. SIDER contiene, infatti, tutti gli effetti collaterali documentati sui farmaci in commercio, che hanno una frequenza di riscontro nei diversi campioni variabile. Molte reazioni avverse sono estremamente rare e riscontrate poche volte.

Per il corretto addestramento della rete neurale, è essenziale fornire un dataset contenente un congruo numero di esempi per ogni categoria. Ad esempio imponendo di prendere gli effetti collaterali univoci che si verificano in un numero minimo di 10 farmaci, si passa da un totale di 4251 effetti collaterali univoci, distribuiti su 1556 farmaci diversi, ad un totale di 1389 effetti collaterali diversi, con un taglio percentuale di circa il 67,32%, il che dimostra come molti effetti collaterali nel dataset siano poco comuni.

I risultati della rete sono stati analizzati con diverse combinazioni di filtraggio sul dataset, esaminando le prestazioni di predizione a partire dagli effetti collaterali più rari, fino all'elaborazione di solo gli effetti collaterali più comuni.

Infine, i dati di target per la rete Multilayer Perceptron, sono stati ulteriormente processati per portarli ad una forma compatibile al problema di classificazione di tipologia multi-label. È stata dunque creata una matrice di dimensioni (c, n) , ordi-

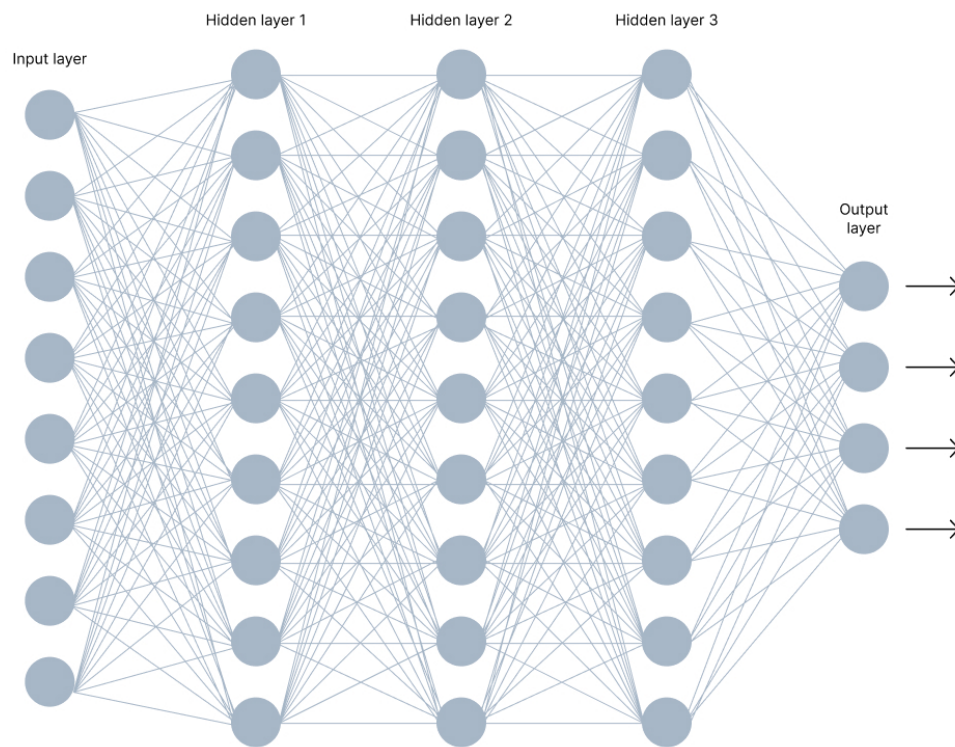
nata in ordine crescente in base agli identificativi dei composti. La dimensione c è il numero di campioni ed n è pari al numero di effetti collaterali univoci. Iterando sulle righe passeremo quindi da un farmaco al successivo. Le colonne costituiscono la totalità degli effetti collaterali univoci, o in altre parole, le possibili classi o "label" assegnabili ad un farmaco. Prendendo il singolo composto farmaceutico (la singola riga), questo ha un vettore binario assegnato: in corrispondenza degli effetti collaterali propri della molecola, è rintracciabile il valore 1. Il valore 0 occuperà le altre posizioni.

3.1.3 Partizione del Dataset

A questo punto dell'elaborazione, il dataset nella sua totalità ha assunto la corretta forma per poter essere effettivamente processato dal modello predittivo. Il passo successivo consiste nella suddivisione dei dati in tre sottogruppi, per il training, la validazione ed il test della rete. La distribuzione percentuale dei campioni si attesta al 70% per il Training Set, che contiene la maggior parte degli esemplari, al 10% per il Validation Set e al 20% per il Test Set.

3.2 Costruzione della Rete Neurale Artificiale e Definizione degli Iperparametri

Nella definizione di un modello di rete neurale, vengono stabiliti dei parametri definiti a priori, detti **iperparametri**. Il prefisso "iper" suggerisce che si tratta di parametri "di primo livello" che controllano il processo di apprendimento e che sono definiti esternamente al modello. I loro valori non possono essere cambiati durante l'addestramento [15] e, pertanto, è importante selezionarli accuratamente (per esempio tramite *grid search*).



V7 Labs

Figura 3.3: Rete neurale composta da strati densamente interconnessi. Ogni arco è caratterizzato da un peso, che viene adattato durante l'apprendimento.

Definiti gli iperparametri, la rete ha ora il compito di adattare i suoi parametri interni, i pesi di interconnessione fra neuroni (vedi Figura 3.3), per trovare la mappatura ottimale tra le caratteristiche strutturali in ingresso (le feature) e gli effetti collaterali, i target, in uscita. La rete viene fatta apprendere sul Training Set.

Per la validazione delle prestazioni con gli iperparametri assegnati, i pesi interni della rete rimangono fissi e uguali a quelli a cui si è giunti al passo precedente. Cambiano invece i dati in ingresso e i valori target, sfruttando questa volta quelli provenienti dal Validation Set. La capacità di predizione della rete verrà infine messa alla prova sul Test Set, costituito da dati non utilizzati durante l'apprendimento: si tratta di un'importante verifica dell'abilità di generalizzazione della rete.

3.2.1 Rete Neurale Artificiale: gli Iperparametri

All'atto pratico è stata applicata una grid search per il dimensionamento degli iperparametri, ricercando tra le diverse configurazioni di:

- numero di strati nascosti ("hidden") nella rete,
- numero di neuroni per strato,
- numero di epoche di apprendimento,
- "step" (o passo) di apprendimento,
- metodo di apprendimento,
- funzione di attivazione.

I primi due parametri sono architetturali e definiscono la struttura e la profondità della rete. Il numero di epoche di apprendimento definisce il numero di iterazioni che il modello deve eseguire, ovvero il numero di volte che il Training Set viene presentato alla rete durante la fase di addestramento. Dopo ogni ciclo la rete neurale aggiusta i suoi parametri. Il passo di apprendimento descrive di quanto variano i pesi ad ogni iterazione, ovvero la "velocità di discesa" lungo la superficie dell'errore. Il metodo di apprendimento definisce il modo in cui i parametri della rete vengono ottimizzati. I due metodi principali per gli MLP sono SGD, "Stochastic Gradient Descent", e "Adam optimizer". Entrambi minimizzano la funzione di errore, tramite discesa sul gradiente. In altre parole stimano i valori ottimi dei parametri, determinando la traiettoria massima di discesa per giungere al minimo della funzione errore. Adam aggiorna questo meccanismo introducendo un dimensionamento dello step di apprendimento adattativo e automatico, per ogni variabile di input. Infine, il ruolo principale della funzione di attivazione è quello di trasformare la somma degli input pesati ad un certo neurone in un valore di uscita da inviare al successivo

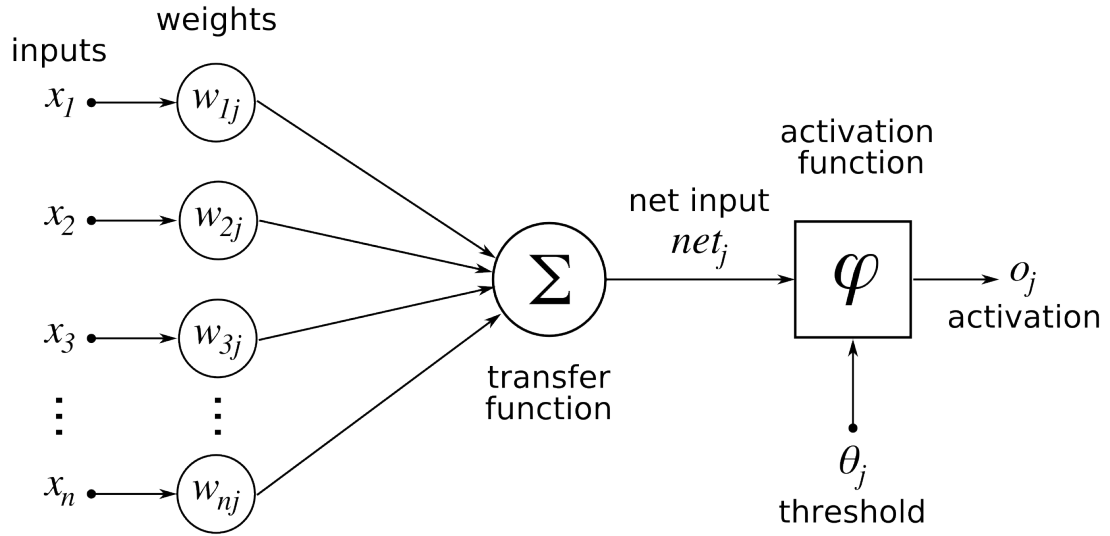


Figura 3.4: Diagramma di un neurone artificiale con somma pesata degli input e funzione di attivazione.

livello o in output. Lo scopo della funzione di attivazione è quello di aggiungere non linearità alla rete neurale. Supponendo di avere una rete neurale senza le funzioni di attivazione, ogni neurone eseguirà solo una trasformazione lineare sugli input, utilizzando i pesi [18]. D'altro canto una rete neurale con neuroni dotati di una funzione di attivazione non lineare può approssimare una qualsiasi funzione reale, limitata e continua.

3.2.2 Rete Neurale Artificiale: Implementazione

La ricerca degli iperparametri è stata implementata mediante cicli *for* innestati. Ciascun ciclo *for* esegue l'iterazione su un vettore di possibili configurazioni per l'iperparametro analizzato.

Lo strato di output è l'ultimo livello della rete, che porta le informazioni apprese attraverso il livello nascosto e fornisce di conseguenza il valore finale [18]. L'ultimo strato di neuroni è stato reso esente dalle variazioni nella configurazione degli iperparametri, ovvero lo strato di output e la sua funzione di attivazione non variano

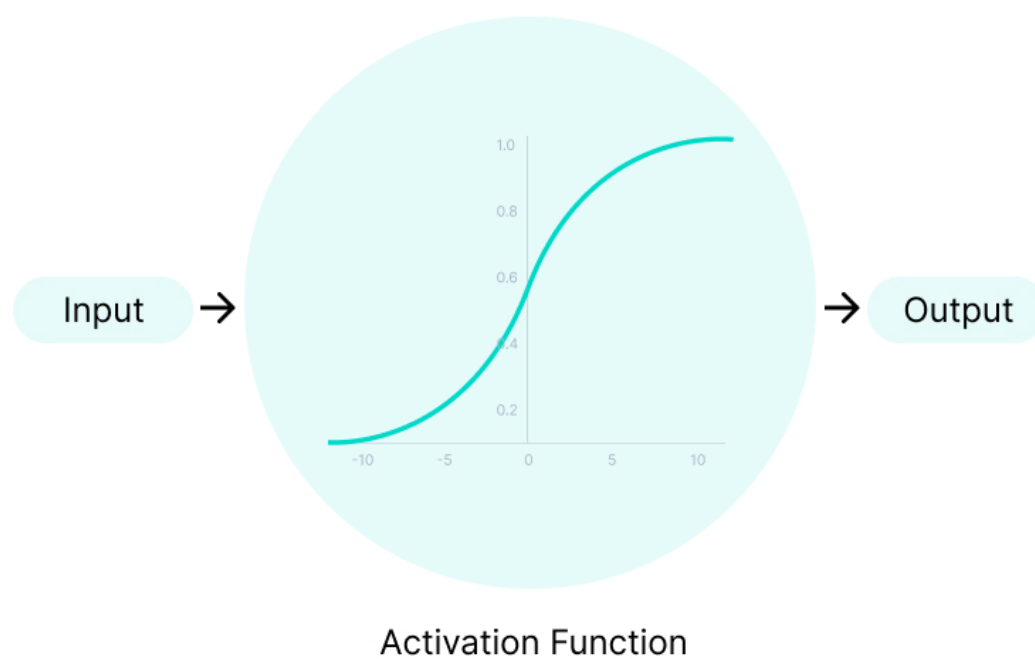
con le diverse configurazioni. Il motivo è da ricercare nella tipologia di problema che andiamo a risolvere: all'uscita della rete, vogliamo un vettore che ha un numero di componenti pari al numero di classi (effetti collaterali) possibili, per ogni campione (farmaco) in ingresso. L'ultimo strato ha di conseguenza un numero di neuroni pari esattamente al numero di classi. Inoltre le classi (effetti collaterali) sono indipendenti l'una dall'altra. È dunque utile avere una funzione di attivazione che produce in output un vettore di n valori compresi tra 0 e 1, dove n rappresenta il numero di classi, 1 rappresenta la certezza che il campione (farmaco) appartenga alla data classe (produca quell'effetto collaterale) e 0 la certezza che non lo produca. Questo ruolo viene gestito bene dalla funzione di attivazione logistica, o sigmoide (Figura 3.5) descritta da:

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

Al valore di output è poi possibile applicare una soglia sopra la quale la classe risulta positiva, e viceversa sotto la quale gli assegneremo il valore nullo.

3.2.3 Il Problema delle Classi Sbilanciate

Un problema di classificazione sbilanciato si ha quando la distribuzione degli esempi tra le classi note è disomogenea. Nel problema trattato, l'assenza di un effetto collaterale è molto più frequente della sua presenza. Le classificazioni sbilanciate rappresentano una sfida per la modellazione predittiva, poiché la maggior parte degli algoritmi di apprendimento automatico sono stati progettati in base al presupposto di avere un egual numero di esempi per ciascuna classe. Ciò si traduce in modelli con scarse prestazioni predittive, in particolare per la classe di minoranza. Questo è un problema perché tipicamente la classe di minoranza è più importante e il modello



V7 Labs

Figura 3.5: Funzione di attivazione logistica. Questa funzione accetta qualsiasi valore reale come input e genera valori nell'intervallo da 0 a 1.

acquisisce una maggiore sensibilità agli errori di classificazione per questa classe [19]. Per ovviare a questo problema ho calcolato un vettore di pesi, dove ogni componente del vettore assegna un valore a ciascun campione del Training Set, proporzionale al numero di classi positive assegnate. Maggiore il numero di etichette di un farmaco e maggiore sarà il suo peso nell'apprendimento.

3.2.4 Libreria di Compilazione

Per la compilazione della rete Multilayer Perceptron nel suo complesso, con gli iperparametri e il vettore di pesi assegnati, ho utilizzato il modello Sequential della libreria Keras, l'API di alto livello di TensorFlow 2 [17].

Capitolo 4

Risultati

Nel presente capitolo verranno descritti i risultati ottenuti, comprensivi della scelta dei parametri di valutazione delle prestazioni.

4.1 Metodi di Valutazione

Un metodo di valutazione molto popolare e intuitivo consiste nel calcolo dell'accuratezza della rete. L'accuratezza misura la frequenza con cui un modello classifica correttamente i dati. Generalmente un modello è considerato "buono" se è "accurato". Tuttavia sui dataset sbilanciati, l'accuratezza può non costituire il miglior modo di valutare le prestazioni di una rete [20]. Si consideri il caso in cui i dati appartengano a due classi, dove la prima classe è molto più rappresentata della seconda. In questa situazione, se la rete predice in uscita unicamente valori appartenenti alla prima classe, avrà comunque un livello di accuratezza elevato. Facendo un esempio con 10 elementi, di cui 9 appartengono alla prima classe e solo 1 alla seconda, se il modello categorizza tutti gli esempi come appartenenti alla prima classe, l'accuratezza sarà pari a 0.9. Il risultato è un punteggio molto elevato, ma che attesta anche la totale incapacità di predizione verso la classe di minoranza.

Per avere una visione più precisa dei risultati di predizione si è deciso di utilizzare la **matrice di confusione**.

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

Gli elementi classificati correttamente possono essere del tipo:

- "True Positive" (TP): previsione corretta della classe positiva.
- "True Negative" (TN): previsione corretta della classe negativa (nel caso specifico, l'effetto collaterale non è proprio del farmaco).

Gli elementi che risultano classificati erroneamente dalla rete saranno:

- "False Positive" (FP), nel caso sia stata fatta una previsione errata della classe positiva.
- "False Negative" (FN), se la previsione della classe negativa risulta errata.

Da questa tabella possiamo ottenere la misura di "**precision**" (precisione), "**recall**" (richiamo) e "**f1 score**" (punteggio f1).

La precisione è il rapporto $\frac{TP}{TP+FP}$ e, intuitivamente, rappresenta la capacità del classificatore di non etichettare una classe come positiva, per un campione che non possiede tale proprietà.

Il richiamo è il rapporto $\frac{TP}{TP+FN}$. Quando un modello è sensibile per una classe, la prevede ogni volta che si verifica.

Per entrambi gli score il valore migliore è 1 e il valore peggiore è 0.

Il punteggio F1 può essere interpretato come una media armonica della precisione

e del richiamo, dove un punteggio F1 raggiunge il suo valore migliore a 1 e il punteggio peggiore a 0. Il contributo relativo di precisione e richiamo al punteggio F1 è uguale [21]. La formula per il punteggio F1 è $2 \times \frac{\text{precisione} \times \text{richiamo}}{\text{precisione} + \text{richiamo}}$

Un altro tra i metodi valutativi della qualità della predizione è la curva di precisione-richiamo. È una curva che combina precisione e richiamo in un'unica visualizzazione, calcolata per ogni soglia. Più alta è la curva sull'asse y, migliori sono le prestazioni del modello [25]. È possibile ottenere un "riassunto" della curva in un unico numero, calcolando la media ponderata delle precisioni raggiunte a ciascuna soglia, "average precision" (precisione media), valutata matematicamente come

$$\sum_n (R_n - R_{n-1}) P_n$$

P_n e R_n sono la precisione il richiamo alla n -esima soglia e R_{n-1} è il richiamo calcolato alla soglia precedente. Il termine tra le parentesi è utilizzato come peso della misura della precisione.

Infine per la validazione del modello è stata utilizzata anche la curva ROC (Receiver Operating Characteristic) e il punteggio AUC (Area Under the Curve).

La ROC è un grafico che visualizza il compromesso tra tasso di veri positivi (TPR) e tasso di falsi positivi (FPR), calcolato e tracciato per ogni soglia.

La precisione media è più sensibile ai miglioramenti predittivi che riguardano la classe positiva, rispetto al punteggio AUC. Questo è ben evidenziato dalla definizione stessa di precisione e richiamo.

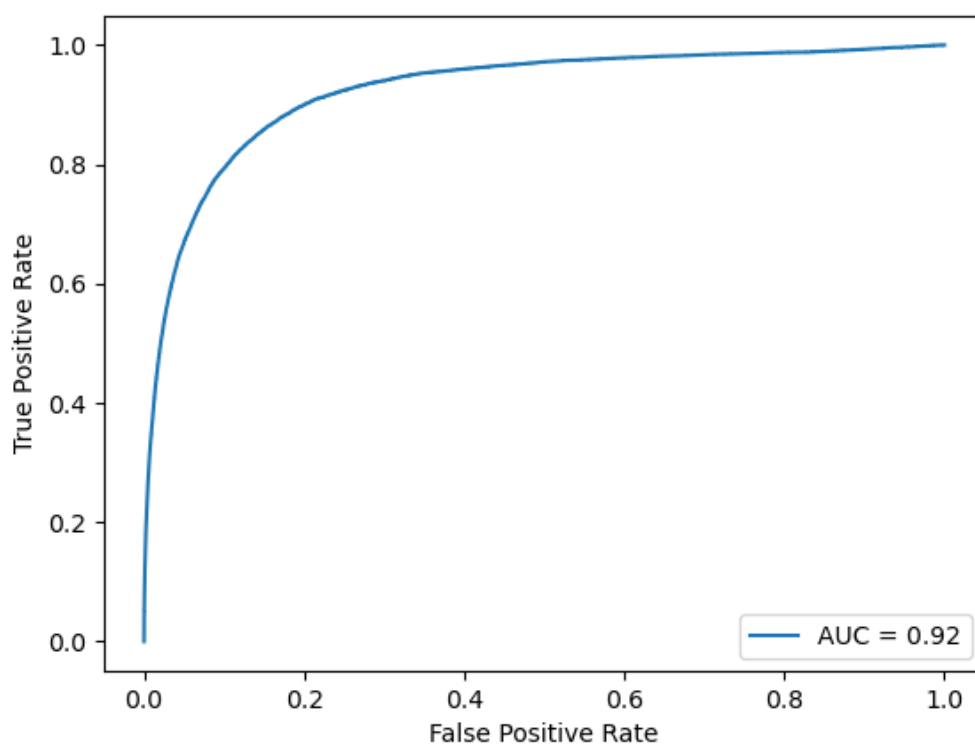


Figura 4.1: Esempio di curva ROC. Nessun tetto massimo né minimo è stato impostato per il filtro sulle occorrenze degli effetti collaterali.

Iperparametri della rete: neuroni nello strato nascosto = (800), funzione di attivazione = sigmoide, metodo di apprendimento = Adam, step iniziale di apprendimento = 0.0001, numero di epoche di apprendimento = 400. Curva tracciata su Validation Set, prima della validazione finale sul Test Set.

4.2 Prestazioni del Modello Predittivo

Le migliori prestazioni in termini di punteggio AUC sono state ottenute senza applicare nessuno filtro sul numero di occorrenze degli effetti collaterali. In questo caso viene dato un egual peso alla predizione delle etichette positive e negative da parte del modello.

Il totale di effetti collaterali univoci da predire è pari a 4251 e sono presenti 1505 molecole farmaceutiche diverse. Per come è strutturato il vettore target, inevitabilmente, sono presenti molti più valori 0 che 1. Infatti, ad ogni molecola è affiancato un vettore target con un numero di elementi pari al totale degli effetti collaterali univoci. I valori 1 sono presenti solo in corrispondenza degli effetti collaterali propri della molecola. Poiché la concomitanza di molti effetti congiunti è rara, i vettori target sono dominati dai valori nulli.

Come descritto in precedenza, la soluzione applicata, che aggira in piccola parte il problema, è stata quella di calcolare un vettore di pesi che include una costante per ogni campione molecolare. Le componenti del vettore è proporzionale al numero di effetti collaterali nel campione stesso. In questo modo, durante l'apprendimento, la rete dà più peso ai campioni con molti effetti collaterali, non favorendo eccessivamente la classe di maggioranza.

I seguenti iperparametri, ottenuti ottimizzando la precisione media (AP), sono risultati i migliori nei vari test: funzione di attivazione sigmoide, metodo di ottimizzazione Adam, step di apprendimento iniziale di 10^{-4} — e sono stati mantenuti costanti nei diversi test. Tuttavia differiscono gli iperparametri riguardanti il numero di neuroni nascosti e il numero di epoche di apprendimento, che verranno riportati in tabella. Dai risultati in Tabella 4.1 possiamo vedere come applicare i pesi che modificano la rilevanza dei campioni durante l'apprendimento abbia apportato un lieve miglioramento nei risultati.

Peso dei Campioni	Min Occ.	Max Occ.	Unità Nascoste	Epoche	AUC	AP
no	nessun filtro	nessun filtro	1000	300	0.9125	0.3593
sì	nessun filtro	nessun filtro	700	500	0.9181	0.3602

Tabella 4.1: Risultati di predizione sul Test Set del miglior modello. Nessun filtro sulle occorrenze. Numero di effetti collaterali univoci: 4251

Peso dei Campioni	Min Occ.	Max Occ.	Unità Nascoste	Epoche	AUC	AP
no	10	nessun filtro	300	500	0.8395	0.3765
sì	10	nessun filtro	200	600	0.8450	0.3853

Tabella 4.2: Risultati di predizione sul Test Set del miglior modello. Minimo di 10 occorrenze per gli effetti collaterali. Numero di effetti collaterali univoci: 1389

Applicando un filtro sul numero minimo di occorrenze degli effetti collaterali pari a 10, ovvero prendendo gli effetti collaterali che vengono riscontrati in 10 farmaci diversi o più, vengono tagliati via gli effetti collaterali più rari. Il taglio si assesta al 67,32% del totale. Gli effetti collaterali univoci utilizzati per l'apprendimento passano da 4251 a 1389. In questo caso lo sbilanciamento tra le due classi è minore. La Tabella 4.2 mostra come il punteggio AUC diminuisce, ma aumenta la precisione media AP.

Gli ultimi risultati riportati in Tabella 4.3 riguardano la casistica in cui si voglia predire solo i 100 effetti collaterali più frequenti. È stato applicato un filtro sulle occorrenze degli effetti avversi per un riscontro minimo di 370 farmaci. Eliminando dal dataset le molecole che presentavano solo gli effetti collaterali più rari, il totale passa dalle precedenti 1505 molecole a 1493 campioni molecolari.

In appendice, è possibile trovare i risultati per altre configurazioni dei parametri. Infine, l'ultima considerazione riguarda la soglia per la predizione delle etichette. Solitamente le predizioni vengono calcolate discriminando se la classe appartiene o meno al campione in base alla soglia predefinita di 0.5. Modificando questa soglia

Peso dei Campioni	Min Occ.	Max Occ.	Unità Nascoste	Epoche	AUC	AP
no	370	nessun filtro	100	300	0.6849	0.5910
sì	370	nessun filtro	100	300	0.6865	0.5931

Tabella 4.3: Risultati di predizione sul Test Set del miglior modello. Minimo di 370 occorrenze per effetti collaterali. Numero di effetti collaterali univoci: 100

è possibile avere risultati predittivi differenti. Abbassando la soglia, ad esempio, aumenteremo la probabilità che il modello applichi l’etichetta dell’effetto collaterale al farmaco. Questo implica un aumento del punteggio di richiamo, e, inevitabilmente, una conseguente diminuzione nel punteggio di precisione. Può risultare un buon compromesso nel caso l’obiettivo risulti predire il quantitativo maggiore di effetti collaterali del farmaco, prima di un susseguente studio clinico.

Per l’ultima configurazione in Tabella 4.3, il miglior punteggio $F1$, con un valore di 0.5924, è stato ottenuto applicando una soglia pari a 0.15. La *precisione* si attesta a 0.4749 e il *richiamo* ha il valore di 0.7874. Un richiamo abbastanza elevato, che mostra una predizione finale di buona parte degli effetti collaterali.

Per la soglia a 0.5, invece, si documentano: $F1 = 0.5352$, *precisione* = 0.5839, *richiamo* = 0.4940.

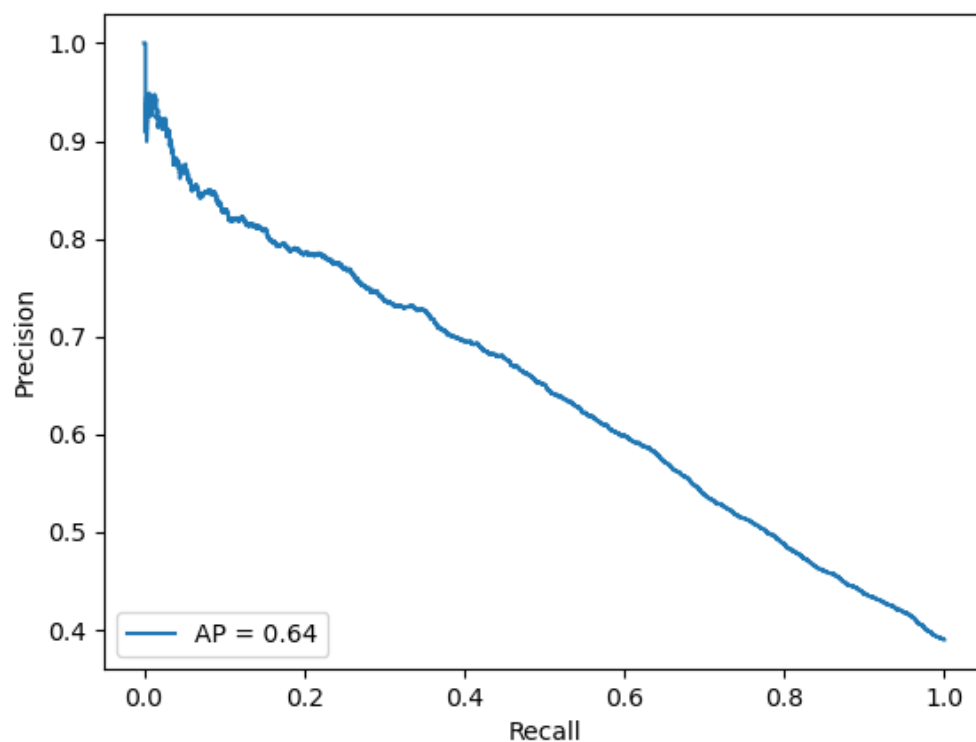


Figura 4.2: Esempio di curva precisione-richiamo. Viene impostato un minimo di 367 e un massimo di 1500 occorrenze per gli effetti collaterali.

Iperparametri della rete: neuroni nello strato nascosto = (300), funzione di attivazione = sigmoide, metodo di apprendimento = Adam, step iniziale di apprendimento = 0.0001, numero di epoche di apprendimento = 300. Curva tracciata su Validation Set, prima della validazione finale sul Test Set.

Capitolo 5

Conclusioni

Il processo di sviluppo delle molecole a scopo farmaceutico è un processo lungo e costoso. Gli effetti avversi imprevisti, che si verificano durante il processo di sviluppo del farmaco, possono sospendere l'intero prosieguo. Pertanto, la previsione a priori degli effetti collaterali del farmaco, in fase di progettazione, è fondamentale. In questo progetto, è stata utilizzata la struttura chimica del farmaco e i fingerprint delle molecole per prevedere le reazioni collaterali avverse, ottenendo risultati promettenti.

Appendice

Qualche esempio di funzione

Il progetto è consultabile in versione integrale su <https://github.com/carlomerola/Adverse-Drugs-Effects-Prediction>. In questa, sezione si riportano solo alcune delle funzioni implementate, spiegandone l'impiego. Il codice è stato scritto in linguaggio Python. Le librerie utilizzate sono state: numpy, pandas, matplotlib, sklearn, rdkit, tensorflow.

Main

Nella funzione main vengono definiti i percorsi e le variabili da passare alle funzioni. Inoltre vengono istanziate le classi e chiamate le funzioni stesse. Pertanto dal codice sottostante è possibile estrapolare una visione generale delle implementazioni.

```
def main():

    matplotlib.use('Agg')

    #-- get the current directory and define paths/variables --
    current_directory = os.path.dirname(os.path.abspath(__file__))
    pbc_path = os.path.join(current_directory,
                             'Data/PubChem.compound.stereocid.csv')
    se_path = os.path.join(current_directory, 'Data/SIDER/meddra.all.se.tsv')
    o_cut=os.path.join(current_directory, 'Output/filtered-side-effects.info.txt')
    o_se = os.path.join(current_directory, 'Output/output-se.csv')
```

```

o_feat = os.path.join(current_directory, 'Output/output_feat.csv')
o_scaled = os.path.join(current_directory, 'Output/output_scaled_feat.csv')
o_dummies = os.path.join(current_directory, 'Output/output_dummies.csv')
o_scores = os.path.join(current_directory, 'Output/output_scores.txt')
o_pr = os.path.join(current_directory, 'Output/PR_Curve_Plots/')
o_roc = os.path.join(current_directory, 'Output/ROC_Curve_Plots/')

#-- filter side effects by term and occurencies --
meddra_type = 'PT' #-- MedDRA concept type.
                #   Preferred Term (PT) or Lowest Level Term (LLT) --
min_occ = 10      #Min occurancies of a side effect for acceptance
max_occ = 800     #Max occurancies of a side effect for acceptance

#-- features options --
fp_size = 2048    #Default fingerprint size is 2048
get_only_fp = False #Get only fingerprints or also PubChem descriptors?

#-- initiate class Dataset and call class functions --
dataset = DataSet(pbc_path, se_path)
pt_df = dataset.FilterSEByType(meddra_type)
filtered_se = dataset.FilterSEByOcc(min_occ, max_occ, pt_df, o_cut)
drugs_se_dict = dataset.GetDict(filtered_se, pt_df)
features_df = dataset.GetFeatures(drugs_se_dict, fp_size)
drugs_se_dict, features_df = dataset.Consistency(drugs_se_dict, features_df)

#-- print to file --
dataset.PrintDataFrame(drugs_se_dict, o_se)
dataset.PrintDataFrame(features_df, o_feat)

y, check = dataset.GetDummies(drugs_se_dict, filtered_se)

#-- print to file --
dataset.PrintDataFrame(check, o_dummies)

X = dataset.ScaleAndVectorizeFeatures(features_df, get_only_fp)

#-- print to file --
dataset.PrintDataFrame(X, o_scaled)

#-- initiate class MultiLabelClassifier and call class functions --
classifier = MultiLabelClassifier(X, y)
X_train, X_test, X_val, y_train, y_test, y_val, sweights, cweights
= classifier.SplitAndWeight()
classifier.ModelEvaluation(X_train, X_test, X_val, y_train, y_test, y_val,
                           sweights, cweights, o_scores, o_pr, o_roc)

```

Matrice binaria degli effetti collaterali

A questa funzione vengono passati il dizionario che associa, ad ogni identificativo molecolare, i suoi effetti collaterali e gli effetti collaterali filtrati sia per termine Med-DRA "PT", sia per numero di occorrenze degli effetti collaterali. Queste strutture sono state calcolate in precedenza e non vengono qui riportate.

La funzione ritorna la matrice binaria degli effetti collaterali con dimensione (c, n) . c risulta uguale al numero di campioni, n uguale al numero di effetti collaterali univoci.

```
def GetDummies(self, CID_SE_, filtered_se_):

    -- Create unique side effects columns.
    # Populate with ones only SE of corresponding drug --
    all_filtered_side_effects = list( filtered_se_.keys() )

    df = pd.DataFrame(0., index=CID_SE_.keys(),
                      columns = all_filtered_side_effects )

    for drug in CID_SE_.keys():
        for side_effect in CID_SE_[drug]:
            df.loc[drug, side_effect] = 1.

    -- check --
    print("\nThe number of columns in Dummies table is: %d\n" %df.shape[1])
    print(f'\nSorted side effects in dummies: \n{df}')

    check = df
    y = df.to_numpy()

    return y, check
```

Matrice delle caratteristiche non standardizzate

Nella funzione vengono estratte le caratteristiche chimiche di interesse, prelevate dal database PubChem. Dopodiché vengono estratti i fingerprint, a partire dalle stringhe SMILES, e sostituiti con le stesse.

In una funzione successiva, le caratteristiche sono state standardizzate e portate ad avere la struttura adeguata all'elaborazione.

```
def GetFeatures(self, CID_SE_, fpSize_):
    df = pd.read_csv(self.pubchem_path, index_col = ['cid'],
        usecols = ['cid', 'mw', 'polararea', 'xlogp', 'heavycnt',
        'hbonddonor', 'hbondacc', 'rotbonds', 'isosmiles'])

    #-- include only the rows in the df dataframe that have a 'cid' value
    #   which is also found in CID_SE_.keys() --
    df = df[df.index.isin(CID_SE_.keys())]

    #-- set null features to 0 --
    df.fillna(0, inplace=True)

    print("\n\nFeatures:\n")
    print(df)

    #-- iterate over rows to extract fingerprints --
    for index, row in df.iterrows():
        smiles = row['isosmiles']
        mol = Chem.MolFromSmiles(smiles)
        if mol is not None:
            fp = rdkit.Chem.rdmolops.RDKFingerprint(mol, fpSize = fpSize_)

            #--Transforming RDKit ExplicitBitVect() into binary numpy array--
            binary = np.zeros(len(fp), dtype = float)
            for i in range( len(fp) ):
                if fp[i] > 0:
                    binary[i] = 1
            df.at[index, 'isosmiles'] = binary

        else:
            print(f'Invalid SMILES: {smiles}')

    #-- rename isosmiles column to fingerprints column --
    df.rename( columns = {'isosmiles': 'binary_fingerprints'}, inplace = True)

    #-- check --
    print("\n\nFeatures_with_derived_fingerprints:\n")
    print(df)

    return df
```

Valutazione della coerenza dei dati

I dati sono stati prelevati da fonti diverse. Questa funzione si assicura che i vettori "features" e "target" siano ordinati secondo lo stesso ordine degli identificativi molecolari. Inoltre, rimuove i farmaci che non sono presenti in entrambe le strutture dati.

```
def Consistency(self, cid_se_dict_, cid_feat_df_):

    #-- stereo-cids need to coincide --
    se_sorted = dict(sorted(cid_se_dict_.items() ) )
    feat_sorted = cid_feat_df_.sort_index()

    keys = list(se_sorted.keys())
    indexes = list(feat_sorted.index)

    rmvd_drugs = []

    i = 0
    while i < len(indexes):
        if indexes[i] not in keys:
            rmvd_drugs.append( indexes[i] )
            del feat_sorted[ indexes[i] ]

        i += 1

    i = 0
    while i < len(keys):
        if keys[i] not in indexes:
            rmvd_drugs.append( keys[i] )
            del se_sorted[ keys[i] ]

        i += 1

    print(f'\nRemoved_drugs: {rmvd_drugs}')

    #-- check --
    new_keys = list(se_sorted.keys())
    new_indexes = list(feat_sorted.index)
    print("\nEvaluating_data_consistency:")
    print("Cid_SE_rows: %d" % len(se_sorted.keys() ) )
    print("Cid_Feat_rows: %d" % feat_sorted.shape[0] )

    if ( new_keys != new_indexes ):
        print("\nDrugs_in_features_and_targets_aren't_the_same!")

    return se_sorted, feat_sorted
```

Una parte della funzione di valutazione del modello

Alla funzione vengono passati i 3 sotto-insiemi di Training, Validation e Test, che vengono utilizzati per l'apprendimento del modello. Qui sotto viene riportato il codice presente all'interno del loop per la valutazione degli iperparametri migliori, stimando i punteggi sul Validation Set.

Successivamente, per la valutazione finale delle prestazioni, il modello migliore verrà testato sul Test Set. Inoltre verrà trovata la migliore soglia che ottimizza il punteggio F1.

```
#-- Repeating the following code into nested for loops
#   to iterate over all combinations of hyperparameters --

count += 1
print("#%d" %count)

current_params = {'hidden_layer_sizes': hls,
                  'activation': activ_fun, 'solver': solver_,
                  'learning_rate': lr,
                  'learning_rate_init': init,
                  'max_iter': epochs
}

#-- Create keras sequential layers.
#   Last layer activation function is always sigmoid --
model = Sequential()

for i, layer in enumerate(current_params['hidden_layer_sizes'] ):
    if i == 0:
        model.add( Dense(units = layer, activation = current_params['activation'],
                        input_shape = (X_train.shape[1], ) ) )
    else:
        model.add(Dense(units = layer, activation = current_params['activation']))

#-- add last layer with units as number of target classes --
model.add( Dense( units = y_train.shape[1], activation = 'sigmoid' ) )

if solver_ == 'adam':
    model.compile( optimizer = Adam(learning_rate =
                                   current_params['learning_rate_init']),
                  loss = 'binary_crossentropy' )
elif solver_ == 'sgd':
```

```

        model.compile( optimizer = SGD(learning_rate =
                                current_params['learning_rate_init']),
                        loss = 'binary_crossentropy' )
    else:
        model.compile( loss = 'binary_crossentropy' )

model.fit(X_train, y_train, sample_weight = sweights_,
          epochs = current_params['max_iter'], batch_size = 64)

y_proba = model.predict(X_val)
y_pred = (y_proba >= threshold).astype(int)

print(f'\nPred. Sizes: {y_pred.shape}')
print(y_proba)

precision = metrics.precision_score(y_val, y_pred, average = 'micro')
recall = metrics.recall_score(y_val, y_pred, average = 'micro')
f1 = metrics.f1_score(y_val, y_pred, average = 'micro')

avg_prec = metrics.average_precision_score(y_val, y_proba, average = 'micro')
auc = metrics.roc_auc_score(y_val, y_proba, average = 'micro')

#-- tn, fp, fn, tp --
cm = metrics.multilabel_confusion_matrix(y_val, y_pred, samplewise = True)
print(cm)

current_scores = { 'f1': f1, 'precision': precision, 'recall': recall,
                   'average_precision': avg_prec, 'auc': auc }

print(f'\nParameters set to {current_params}, threshold: {threshold}')
print(f'\nScores: {current_scores}\n')

#-- write to file current values --
f = open(o_scores_, 'a')
f.write(f'\n\n\n#{count}')
f.write(f'\nParameters: {current_params}')
f.write(f'\nScores: {current_scores}')
f.close()

#-- Plot the micro-averaged Precision-Recall curve --
precision_micro, recall_micro, thpr = metrics.precision_recall_curve(y_val.ravel(),
                                                                    y_proba.ravel())
displaypr = metrics.PrecisionRecallDisplay(precision = precision_micro,
                                           recall = recall_micro,
                                           average_precision = current_scores['average_precision'])

displaypr.plot()
plt.savefig(o_pr+'pr'+str(count)+'.png')

#-- Plot micro-averaged Precision-Recall curve --
fpr_, tpr_, throc = metrics.roc_curve(y_val.ravel(), y_proba.ravel())

```

```
displayroc = metrics.RocCurveDisplay( fpr = fpr_, tpr = tpr_, roc_auc = auc)

displayroc.plot()
plt.savefig(o_roc+'roc'+str(count)+'.png')

#-- Update the best combination of hyperparameters --
if current_scores[evaluation1] > best_scores1[evaluation1]:
    best_params1.update(current_params)
    best_scores1.update(current_scores)
    best_threshold1 = float(threshold)
    position1 = int(count)

if current_scores[evaluation2] > best_scores2[evaluation2]:
    best_params2.update(current_params)
    best_scores2.update(current_scores)
    best_threshold2 = float(threshold)
    position2 = int(count)
```


Sitografia

- [1] About SIDER, <http://sideeffects.embl.de/about/>
- [2] About PubChem, <https://pubchem.ncbi.nlm.nih.gov/docs/about>
- [3] In cosa consiste il Machine Learning? <https://www.oracle.com/it/artificial-intelligence/machine-learning/what-is-machine-learning/>
- [4] How Do Machine Learning Algorithms Differ From Traditional Algorithms? <https://analyticsindiamag.com/how-do-machine-learning-algorithms-differ-from-traditional-algorithms/>
- [5] Difference Between Soft Computing and Hard Computing, <https://www.tutorialspoint.com/difference-between-soft-computing-and-hard-computing>
- [6] Artificial Neural Network, https://en.wikipedia.org/wiki/Artificial_neural_network
- [7] Multiclass and multioutput algorithms, <https://scikit-learn.org/stable/modules/multiclass.html>
- [8] SIDER Format Description, <http://sideeffects.embl.de/media/download/README>
- [9] STITCH Chemical Set, <http://stitch.embl.de/download/README>
- [10] Unified Medical Language System (UMLS), <https://www.nlm.nih.gov/research/umls/index.html>
- [11] MedDRA Hierarchy, <https://www.meddra.org/how-to-use/basics/hierarchy>
- [12] La terminologia MedDra, <https://salute.regione.emilia-romagna.it/normativa-e-documentazione/convegni-e-seminari/corsi-di-formazione/seminario-multitematico-di-farmacovigilanza-bologna-18-giugno-2010/la-terminologia-meddra-dalla-teoria-alla-pratica>

- [13] Scikit Learn Robust Scaler, <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html>
- [14] The RDKit Book, https://www.rdkit.org/docs/RDKit_Book.html
- [15] Parameters and Hyperparameters in Machine Learning and Deep Learning, <https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac>
- [16] The Sequential model, https://keras.io/guides/sequential_model/
- [17] About Keras, <https://keras.io/about/>
- [18] Activation Functions in Neural Networks, <https://www.v7labs.com/blog/neural-networks-activation-functions>
- [19] A Gentle Introduction to Imbalanced Classification, <https://machinelearningmastery.com/what-is-imbalanced-classification/>
- [20] Evaluating Multi-label Classifiers, <https://towardsdatascience.com/evaluating-multi-label-classifiers-a31be83da6ea>
- [21] Scikit-Learn Metrics, <https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics>
- [22] Scikit-Learn Precision Score, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#sklearn.metrics.precision_score
- [23] Scikit-Learn Recall Score, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score
- [24] Scikit-Learn F1 Score, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score
- [25] F1 Score vs ROC AUC vs Accuracy vs PR AUC, <https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc>