UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI

Corso di Laurea Magistrale in Informatica

ANALISI DI DATI MULTI-OMICI PER LA PREDIZIONE DELLA PROGNOSI DI PAZIENTI ONCOLOGICI

Relatore: Prof. Elena Casiraghi
Correlatore: Prof. Dario Malchiodi

Tesi di Laurea di:
Alessandro Beranti
Matr. Nr. 977702

ANNO ACCADEMICO 2021-2022

Alla mia famiglia
Oriella, Andrea e Giuseppe

Ringraziamenti

Un grazie sincero a Oriella, Andrea e Giuseppe che mi hanno sempre supportato. Grazie agli amici che mi sono stati vicini, in particolare ai miei compagni di corso Andrea e Samuele con cui ho condiviso tutto percorso magistrale. Infine un sincero ringraziamento ai miei relatori Elena Casiraghi e Dario Malchiodi. Grazie a tutti.

Indice

Introduzione

Il *machine learning* è sempre più diffuso in molti ambiti diversi. L'idea di poter usare questa tecnologia anche in campo medico è affascinante poiché potrebbe permettere di riuscire a migliorare prevenzione, diagnosi e monitoraggio delle malattie. Grazie all'uso di dati provenienti da pazienti è possibile creare modelli di *machine learning* esattamente con questi scopi. In questo lavoro è stato usato apprendimento supervisionato usando *random forest* come algoritmo di classificazione poiché è noto, in letteratura, per essere uno dei metodi migliori quando si hanno dati eterogenei. L'obiettivo è quello di usare dati provenienti dai geni BRCA per predire la prognosi e diagnosi di pazienti al carcinoma mammario invasivo. In particolare si vuole verificare se l'utilizzo di dati multi-omici riesca ad aumentare l'accuratezza della previsione. I geni BRCA1 e BRCA2 sono geni importanti per la salute dei tessuti della mammella e dell'ovaio. Questi geni sono normalmente responsabili della produzione di proteine che proteggono dalle mutazioni del DNA e prevengono la crescita di cellule anormali. Tuttavia, quando i geni BRCA1 e BRCA2 sono mutati, le proteine che producono non funzionano correttamente e c'è un rischio maggiore di sviluppare tumori del seno e dell'ovaio.

I dati a disposizione sono dati multi-omici, ovvero un insieme che contiene le variazioni molecolari su più livelli quali: genomica, epigenomica, trascrittomica, proteomica, metabolomica e microbiotica. Grazie alla ricerca ci sono stati importanti progressi in diversi campi omici e si è capito che la risposta a una domanda in medicina non è solo da ricercare in un unico tipo di dato poiché questi dati si influenzano tra loro.

Il lavoro è composto da quattro capitoli. Nel Capitolo 1 viene affrontato il concetto di *machine learning* illustrando le varie tipologie di apprendimento possibili, entrando più nel dettaglio per quanto riguarda l'apprendimento supervisionato e,

in particolare, approfondendo i *random forest*, ovvero l'algoritmo di classificazione usato durante gli esperimenti. Vengono inoltre analizzate due soluzioni presenti in letteratura, in cui si è applicato il paradigma *machine learning* per problemi medici. Nel Capitolo 2 viene discussa la fonte e la struttura dei dati. Vengono inoltre esaminate le fasi appartenenti al *preprocessing* che possono essere seguite per rendere i dati più qualitativi. Infine vengono trattate le tecniche di riduzione della dimensionalità utilizzate durante gli esperimenti. Nel Capitolo 3 vengono illustrati i passi di *preprocessing* usati sul *dataset* fornito, il processo di *model selection*, la metrica di *performance* usata e infine tutti gli esperimenti effettuati con i relativi risultati ottenuti. Per ultimo, nel Capitolo 4, vengono discusse le conclusioni sul lavoro ed eventuali possibili sviluppi futuri.

Capitolo 1

Stato dell'arte

1.1 Machine Learning

Il termine *Machine Learning*, anche chiamato apprendimento automatico in italiano, sta a indicare la capacità dei computer di apprendere e adattarsi agli input forniti. Molto spesso tale termine viene utilizzato per intendere l'Intelligenza Artificiale e viceversa ma non sono la stessa cosa: il *Machine Learning* è un sottoinsieme della categoria più ampia chiamata appunto Intelligenza Artificiale.

L'intelligenza artificiale può essere definita come la branca dell'informatica che si occupa dell'automazione del comportamento intelligente [1]. Tuttavia è importante che sia definito in modo chiaro cosa significhi intelligenza. Sebbene la maggior parte di noi sia certa di poter riconoscere un comportamento intelligente quando lo vede, non è certo che chiunque possa avere una definizione chiara di intelligenza per poter valutare un programma informatico come intelligente o meno. L'intelligenza artificiale si concentra sullo sviluppo di sistemi in grado di eseguire compiti per cui è richiesta l'intelligenza umana, come il ragionamento, l'apprendimento, la comprensione del linguaggio naturale e la percezione visiva. L'aspetto "intelligente" dell'intelligenza artificiale si basa sull'abilità che hanno i sistemi di adattarsi e migliorare continuamente in base a nuovi stimoli (dati) che ricevono, imitando, in qualche modo, il processo di apprendimento dell'essere umano. Di seguito vengono illustrati esempi particolarmente recenti di questi sistemi.

- Assistenti virtuali, come *Siri* o *Alexa*, ovvero agenti software in grado di

eseguire azioni o erogare servizi in base a comandi ricevuti in maniera vocale o testuale. Esistono sistemi simili, anche chiamati *Chatbot* [2], che vengono utilizzati nel *Customer Care* aziendale come primo livello di assistenza con il cliente. Si contraddistinguono per la loro capacità di comprensione del tono del dialogo e di memorizzazione delle informazioni raccolte. Essi sono sistemi di raccomandazione che indirizzano le scelte degli utenti in base a informazioni fornite da essi: famosi sono quelli che suggeriscono un acquisto in base a quelli fatti in precedenza.

- Il *Natural Language Processing* (NLP), ramo dell'intelligenza artificiale che riguarda l'informazione espressa nel linguaggio naturale. Si tratta di sistemi che elaborano il linguaggio, con finalità che possono variare dalla comprensione del contenuto, alla traduzione, fino alla produzione di testo in modo autonomo a partire da dati o documenti forniti in input [3].

- La *Computer Vision*, area dell'intelligenza artificiale che si occupa di creare macchine che siano i grado di "vedere". In questo contesto però tale termine significa essere in grado di estrapolare informazioni da una o più immagini per risolvere un particolare compito [4]. I sistemi di *Computer Vision* vengono implementati in una vasta gamma di applicazioni, sia industriali che scientifiche, tra cui sistemi per il controllo di robot e veicoli autonomi, sorveglianza video, diagnostica medica, riconoscimento di persone e oggetti, lettura di testo e misurazione di proprietà geometriche [5].

- La diagnosi medica, sistemi che sfruttano l'intelligenza artificiale per analizzare immagini e usare dati medici in modo da aiutare la diagnostica di malattie.

- L' analisi dei dati, sistemi che utilizzano l'intelligenza artificiale per analizzare grandi quantità di dati e ottenere informazioni utili per prendere decisioni, per esempio in campo aziendale.

Il *Machine Learning* è un campo di ricerca che si concentra sulla costruzione di metodi che "apprendono", ovvero sistemi in grado di migliorare le proprie prestazioni in un compito attraverso l'esperienza [6]. Più precisamente: un programma

informatico si dice che impara dall'esperienza E rispetto a una classe di compiti T e una misura di prestazioni P, se le sue prestazioni nei compiti in T, misurate da P, migliorano con l'esperienza E [6]. Ad esempio, un programma informatico che impara a giocare a dama potrebbe migliorare le sue prestazioni misurate dalla sua capacità di vincere nella classe di compiti di giocare a partite di dama, attraverso l'esperienza ottenuta giocando contro se stesso.

Un algoritmo di *Machine Learning* è una procedura, o una serie di istruzioni, che vengono eseguite sui dati con l'obiettivo di creare un modello. Un modello di *Machine Learning* è il risultato finale dell'applicazione di un algoritmo di *Machine Learning* a un insieme di dati di *training*. Gli algoritmi di apprendimento automatico costruiscono un modello basato sui dati, noti come *training data*, al fine di effettuare previsioni o decisioni senza essere esplicitamente programmati per farlo[1]. Gli algoritmi vengono divisi in quattro categorie distinte a seconda del tipo di dato usato per eseguire la fase di apprendimento. Di seguito sono elencate le categorie:

- supervised learning,
- non supervised learning,
- semi-supervised learning,
- reinforced learning.

1.1.1 Supervised Learning

Nel *Supervised Learning*, anche chiamato apprendimento supervisionato in italiano, si ha un insieme di dati X chiamato *training set* composto da N coppie di *input-output*, $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ dove ogni y_j è stata generata da una funzione sconosciuta $y = f(x)$. Lo scopo è trovare la funzione h che meglio approssima la funzione f [7]. Imparare non è altro che la ricerca, all'interno dello spazio delle possibili ipotesi, della funzione che meglio approssima la funzione di partenza. Per misurare l'accuratezza di un'ipotesi passiamo alla funzione trovata

[1] "Senza essere esplicitamente programmati per farlo" è una definizione attribuibile a Arthur Samuel, colui che coniò il termine *Machine Learning* nel 1959.

un *test set* di esempi diversi dal *training set*. Un *training set* non è altro che un insieme di dati utilizzato per addestrare il modello di *Machine Learning*. Un *test set*, invece, è un insieme di dati che non è stato usato durante la fase di addestramento. Possiamo dire che un modello generalizza bene se prevede correttamente il valore di y per gli esempi che non ha usato durante la fase di addestramento.

Quando l'*output* y è un valore di un insieme finito (come per esempio {soleggiato, piovoso, nuvoloso}), il problema di apprendimento è chiamato classificazione; in particolare si dice binaria o booleana se ci sono solo due valori nell'insieme. Questi tipi di valori si dicono anche categorici, essi sono variabili discrete e di norma sono privi di un ordine. Al contrario, quando y è un numero, come per esempio la temperatura atmosferica, il problema di apprendimento è chiamato regressione. Questo tipo di valore è chiamato numerico e può essere rappresentato da un numero reale continuo come da uno discreto.

All'interno di *supervised learning* un problema ricorrente è l'*overfitting*. È molto facile costruire un modello che si adatta perfettamente al *training set* ma potrebbe essere poi incapace di generalizzare bene su dati che non ha usato durante l'addestramento. Per valutare la capacità di generalizzazione di un modello vengono usati dati provenienti dalla stessa popolazione di dati usati per costruire il modello. Questo viene solitamente fatto utilizzando metodi di ricampionamento di dati, come la *cross-validation* [8]. Di seguito vengono illustrate diverse strategie per effettuare la *cross-validation*.

- *K-fold cross-validation*: i dati vengono divisi in k *fold* o gruppi. Il modello viene addestrato usando $k - 1$ sottoinsiemi, che uniti costituiscono il *training set*. Successivamente il modello è applicato al sottoinsieme rimanente che viene chiamato *validation set* e viene misurata la performance. Questo processo viene ripetuto finché ognuno dei k sottoinsiemi non è stato usato come *validation set*. Infine la *performance* del modello viene calcolata come media delle *performance* ottenute a ogni iterazione. Un esempio è visibile in Figura 1.

- *Leave-one-out cross-validation*: speciale caso della *k-fold cross-validation* in cui $k = n$. È un metodo di validazione di un modello di *Machine Learning* che consiste nel prendere un solo elemento del *dataset* e utilizzarlo come *test*

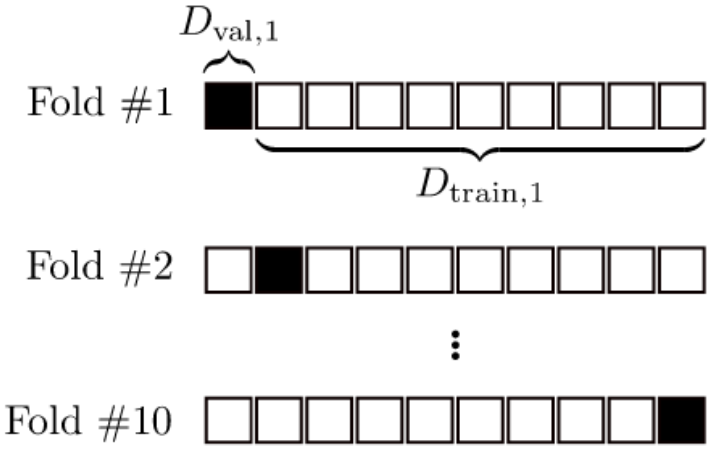


Figura 1: *10-fold cross-validation.* L'insieme di dati, D è randomicamente diviso in 10 sottoinsiemi disgiunti, ognuno dei quali contiene, approssimativamente, il 10% dei dati. Il modello è addestrato usando il *training set* (D_{train}) e successivamente applicato al *validation set* (D_{val}) (fonte: [8]).

set, il resto dei dati viene invece usato come *training set*. Questa operazione viene ripetuta per ogni elemento del *dataset*. In questo modo, ogni elemento viene utilizzato una sola volta come *test set* e il modello viene addestrato sui dati restanti. Il costo computazionale di questa tecnica può essere elevato per grandi insiemi di dati. Tecnica visibile in Figura 2.

Con il termine *overfitting*, o sovradattamento in italiano, si intende l'utilizzo di modelli o procedure che violano il principio di parsimonia, anche noto come rasoio di Occam[2]. In generale, l'*overfitting* si verifica quando un modello è troppo complesso rispetto ai dati disponibili, si adatta quindi troppo bene ai dati di addestramento e non è in grado di generalizzare adeguatamente sui dati nuovi [9].

Un altro aspetto importante nel *machine learning* riguarda le metriche di performance. Esse ci indicano se si stanno facendo progressi nella creazione del modello che meglio si adatta ai dati in *input*. Esistono diverse metriche che possono

[2]Il rasoio di Occam è un principio metodologico che sostiene che, tra le varie spiegazioni possibili per un fenomeno, quella più semplice è probabilmente la più corretta. In altre parole, si dovrebbe scegliere la spiegazione più semplice, tra quelle disponibili, per un fenomeno in quanto essa ha meno probabilità di contenere errori o di essere falsa.

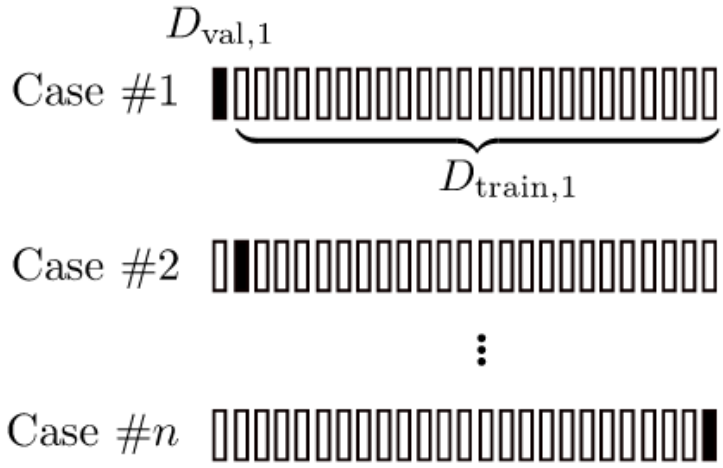


Figura 2: *Leave-one-out cross-validation* applicata su un insieme di dati con $n = 25$ casi. Ogni caso funge da singolo caso di test. Il modello è costruito usando i restanti $n - 1$ casi (fonte: [8]).

essere usate a seconda dei problemi che si stanno affrontando. Se stiamo trattando un problema di regressione, avente quindi *output* continuo, dobbiamo calcolare in qualche modo la distanza tra il dato predetto e quello originale; per fare ciò possiamo usare diverse metriche: *Mean absolute Error (MAE) [10], Mean Squared Error (MSE) [11], Root Mean Squared Error (RMSE) [10],* R^2 *(R-Squared) [12].*

Per quanto riguarda invece un problema di classificazione, essi hanno un *output* discreto quindi abbiamo bisogno di metriche che comparino classi discrete. Le metriche di classificazione valutano le prestazioni di un modello e dicono quanto è buona o cattiva la classificazione. Esistono diverse metriche, di seguito vengono illustrate le più note.

- *Accuracy*: è il rapporto tra un numero di predizione corretto del modello e il numero totale di predizioni effettuate.

- *Precision*: è il rapporto tra il numero di predizioni positive e il numero totale di predizioni positive effettuate dal modello. Questa metrica rappresenta la capacità del modello di non classificare falsamente come positivo un esempio che in realtà è negativo. Intuitivamente esprime la consistenza del risultato.

- *Recall*: anche chiamata *sensitivity* o sensibilità in italiano, è il rapporto tra il numero di predizioni positive corrette (veri positivi TP) e il numero totale di esempi positivi presenti nei dati (TP+FN). Questa metrica rappresenta la capacità del modello di riuscire a identificare correttamente tutti gli esempi positivi.

- *Specificity*: o specificità in italiano, è il rapporto tra il numero di veri negativi (TN) e il numero totale di negativi (TN+TP). In medicina, ad esempio, la specificità di un test può essere utilizzata per determinare la capacità del test di identificare correttamente i pazienti che non hanno una determinata malattia, evitando così il falso positivo.

- *F1-score*: media armonica[3] tra la *precision* e la *recall*. Metrica molto usata quando si vuole mantenere un equilibrio tra *precision* e *recall*, per esempio quando è importante evitare sia falsi positivi che negativi.

- *Area under the Receiver Operating Characteristic curve (AUC-ROC)*: la curva ROC è un grafico che mostra la relazione tra veri positivi e i falsi positivi. L'AUC-ROC è invece una metrica che rappresenta l'area sotto la curva ROC e fornisce una misura generale della capacità del modello di distinguere tra due classi. Un valore di AUC-ROC vicino a 1 indica un modello con prestazioni eccellenti, mentre un valore vicino a 0.5 indica un modello che non è in grado di distinguere tra le classi.

Infine è giusto citare la matrice di confusione anche se non è una metrica. Essa è uno strumento per visualizzare e analizzare i risultati di un modello di classificazione, mostra il numero di osservazioni correttamente classificate e il numero di osservazioni classificato in modo errato. La matrice di confusione è uno strumento per comprendere e interpretare le metriche, ma non è una metrica in sé. In Figura 3 è possibile vedere un esempio reale ottenuto dagli esperimenti effettuati (vedi Paragrafo 3.5).

[3]Dati n numeri, $x_1, x_2, ..., x_n$ la media armonica è data da: $mediaarmonica = \frac{n}{\frac{1}{x_1}, \frac{1}{x_2}, ..., \frac{1}{x_n}}$

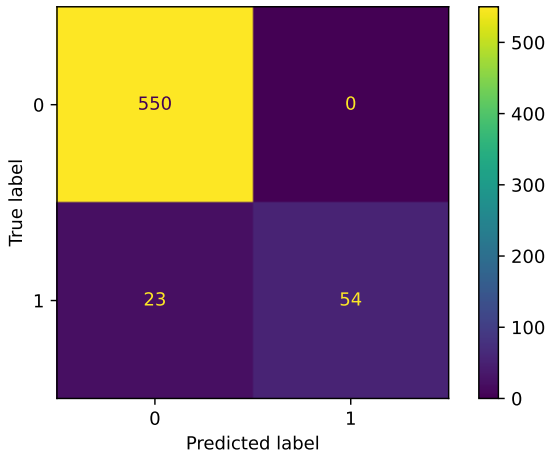


Figura 3: Esempio reale di matrice di confusione calcolata durante gli esperimenti. In particolare, la figura mostra la matrice con la prestazione di AUPRC migliore. È possibile vedere che, per quanto riguarda i pazienti negativi (etichetta 0), il classificatore non sbaglia mai, mentre per quanto riguarda i malati (etichetta 1), il classificatore sbaglia 23 volte su 77. Classifica quindi come sane 23 persone che in realtà sono malate.

Esistono diversi algoritmi di apprendimento supervisionato, ognuno dei quali possiede delle caratteristiche peculiari. Di seguito sono stati approfonditi: *Random Forest*, poiché è l'algoritmo usato durante gli addestramenti, e *Decision Tree*, poiché i primi non sono altro che una foresta dei secondi.

Decision Tree

I *Decision Tree*, o alberi di decisione in italiano, sono algoritmi non parametrici[4] di *supervised learning*. Possono essere utilizzati sia per risolvere problemi di regressione che di classificazione; in particolare in quest'ultima trova maggiore applicazione pratica [13]. Un *Decision Tree* possiede una struttura ad albero, visibile in Figura 4 ed è composta da:

- nodi non terminali: rappresentano un test su una o più caratteristiche,
- ramo: rappresenta un esito del test,

[4]Categoria di algoritmi di *machine learning* che non richiedono la specificazione di un numero fisso di parametri.

- foglia: rappresenta una possibile classe.

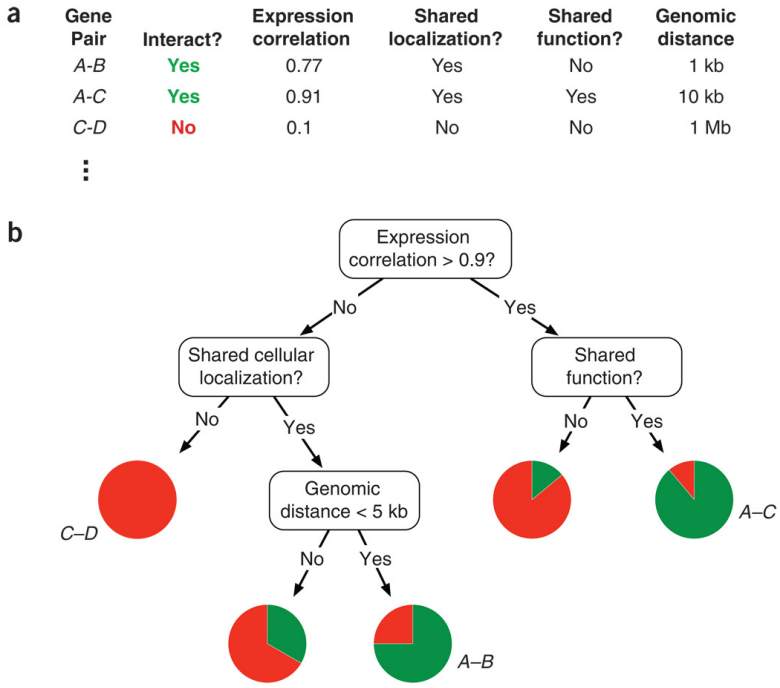


Figura 4: a) Ogni dato è una coppia di geni associata a una serie di caratteristiche. Alcune caratteristiche sono numeri a valore reale (*genomic distance* e *expression correlation*), altre sono categoriche (*shared-localization*). (b) Un ipotetico *Decision Tree* in cui ogni nodo contiene una domanda sì/no che riguarda una singola caratteristica dei dati. Un dato naviga l'albero arrivando a una foglia in base alle risposte alle domande. I grafici a torta indicano la percentuale di interagenti (verde) e non interagenti (rosso) degli esempi di addestramento che raggiungono ciascuna foglia. Si prevede che i nuovi esempi interagiscano se raggiungono una foglia prevalentemente verde o non interagiscano se raggiungono una foglia prevalentemente rossa. (fonte: [14]).

La costruzione di un *Decision Tree* inizia con la scelta della caratteristica più informativa che viene posta come radice dell'albero. Successivamente vengono identificate le sotto-caratteristiche più informative e vengono utilizzate per creare i nodi figli della radice facendo così una suddivisione. La costruzione dell'albero continua su ogni ramo creando ulteriori nodi figli per ogni nuova sotto-caratteristica identificata. La costruzione dell'albero si arresta quando viene raggiunta una delle condizioni di stop:

- tutti i campioni dei nodi terminali appartengono a una stessa classe,
- non ci sono più attributi per un'ulteriore suddivisione.

Per scegliere la caratteristica su cui effettuare una suddivisione vengono comunemente utilizzati due criteri: la *Gini impurity* e l'entropia.

Gini Impurity La *Gini Impurity* misura la probabilità che un elemento scelto a caso, appartenente a una data classe, sia etichettato in modo errato. Un valore basso indica una maggiore purezza delle classi risultanti dalla suddivisione. Consideriamo un dataset D che contiene campioni di k classi diverse. La probabilità che i campioni appartengano alla classe i in un determinato nodo può essere indicata con p_i. La *Gini Impurity* di D può essere definita come:

$$\text{Gini}(D) = 1 - \sum_{k}^{i=1} p_i^2. \tag{1}$$

Il nodo che ha distribuzione di classe uniforme ha l'impurità risultante maggiore. L'impurità minima si ottiene quando tutti i dati appartengono alla stessa classe, quindi l'attributo che genera la minima *Gini Impurity* viene selezionato per dividere il nodo. Se un insieme di dati D viene diviso su un attributo A in due sottoinsiemi D_1 e D_2 con dimensione, rispettivamente, n_1 e n_2, la *Gini Impurity* può essere definita come:

$$Gini_A(D) = \frac{n_1}{n_2} Gini(D_1) + \frac{n_2}{n} Gini(D_2). \tag{2}$$

Nel momento in cui si vuole creare un *decision tree* l'attributo che fornisce la minore *Gini Impurity* ($Gini_A(D)$) viene scelto per suddividere l'albero.

Entropia L'entropia è una misura dell'incertezza associata alle classi. Anche in questo caso, un valore basso indica una maggiore purezza delle classi risultanti dalla suddivisione, più è alta, più l'insieme dei dati è eterogeneo. La formula per calcolare l'entropia è la seguente:

$$Entropia = -\sum(p_i * log_2(p_i)), \tag{3}$$

dove p_i è la probabilità che un dato appartenga alla classe i, la somma viene fatta su tutte le classi presenti nell'insieme di dati. L'entropia è zero quando tutti gli esempi appartengono alla stessa classe e massima quando tutte le classi sono ugualmente rappresentate.

Il criterio di scelta può essere utilizzato per valutare tutti gli attributi disponibili e scegliere quello che produce la suddivisione con il valore più basso. In generale si usano questi criteri per evitare *overfitting* (vedi Paragrafo 1.1.1) e aiutare a generalizzare meglio il modello su dati sconosciuti. Queste tecniche sono euristiche e riescono a trovare un albero efficiente che però non corrisponde necessariamente all'ottimo. La ricerca dell'albero ottimale è computazionalmente difficile perché il numero di possibili alberi cresce molto rapidamente con il numero di attributi. Prendiamo, per esempio, un insieme di dati con un numero di attributi pari a n che ha solo due valori possibili per ogni attributo, ci sono così 2^n possibili alberi di decisione. Se invece ci fossero stati m valori possibili per ogni attributo, allora ci sarebbero stati al massimo m^n possibili alberi di decisione.

La ricerca dell'albero ottimo necessiterebbe quindi dell'esplorazione di tutte queste possibilità e successivamente della valutazione di quale sia l'albero che ha il minor errore sui dati di addestramento. Per questo motivo vengono usati algoritmi *greedy* che creano l'albero prendendo decisioni localmente ottime usando la *Giny Impurity* o l'entropia.

Random Forest

Un *Random Forest*, o foresta aleatoria in italiano, proposto da L. Breiman nel 2001 [15], è un modello matematico di tipo *ensamble*[5] che utilizza un insieme di *Decision Tree*, illustrati nel paragrafo precedente, per fare previsioni. Ciascun *Decision Tree* viene allenato su un sottoinsieme casuale di dati di addestramento e le previsioni finali sono ottenute tramite una votazione di maggioranza o come media delle previsioni degli alberi.

[5] L'*ensamble* è una tecnica di *Machine Learning* che combina più modelli per ottenere una *performance* migliore rispetto a quella ottenuta da ogni modello singolo. Ciò avviene creando diversi modelli di base (ad esempio, *Decision Tree* o reti neurali) su sottoinsiemi differenti dei dati di addestramento e quindi combinando le loro previsioni in una previsione finale.

Un modello di questo tipo risulta più robusto e con una precisione migliore rispetto all'utilizzo di un singolo *Decision Tree*. Esistono diversi motivi per cui si preferisce usare *Random Forest* piuttosto che un singolo *Decision Tree*:

- miglioramento della precisione: usando più *Decision Tree* di solito la precisione finale del modello aumenta,
- riduzione dell'*overfitting*: i singoli *Decision Tree* possono soffrire di *overfitting*. *Random Forest* utilizza allora una serie di *decision tree* e prende la decisione finale come voto di maggioranza. Si può dimostrare che ciò riduce l'effetto dell'*overfitting*,
- rilevamento di *feature* importanti: *Random Forest* permette di individuare quali sono le *feature* più importanti per classificare i dati,
- scalabiltà: *Random Forest* scala molto bene con *dataset* grandi.

È anche possibile usare un *random forest* per fare regressione. In questo caso il *Random Forest* si forma facendo crescere il numero di *Decision Tree* in funzione di un vettore casuale in modo che il predittore assuma valori numerici anziché etichette di classe. I valori di *output* sono numerici e si assume che il *training set* sia estratto in modo indipendente dalla distribuzione del vettore casuale [15].

Questo modello presenta diversi iperparametri i cui valori vengono utilizzati per controllare il processo di apprendimento e di conseguenza le prestazioni del modello rispetto ai dati. Essi sono parametri che non vengono impostati automaticamente dal modello durante l'addestramento ma vengono stabiliti a priori. Tra i più noti troviamo:

- *n_estimators*: numero di alberi nella foresta,
- *max_depth*: profondità massima degli alberi,
- *min_samples_split*: numero minimo di campioni richiesto per dividere un nodo,
- *min_samples_leaf*: numero minimo di campioni richiesto in una foglia,

- *max_features*: numero massimo di caratteristiche considerate per la divisione in ogni nodo,
- *criterion*: funzione per misurare la qualità di una divisione. Alcuni esempi sono: *Gini impurity* ed entropia.

Questi iperparametri possono essere regolati per migliorare le prestazioni del modello e prevenire l'*overfitting*. La scelta ottimale degli iperparametri può essere effettuata utilizzando tecniche di *tuning* (vedi Paragrafo 3.2.1).

1.1.2 Unsupervised Learning

Nel *Unsupervised Learning*, anche chiamato apprendimento non supervisionato in italiano, si ha un insieme di dati X ($\{x_1, x_2, ..., x_n\}$) ma in questo caso non sono presenti né un'etichetta, come invece c'è nel *Supervised Learning* (vedi Paragrafo 1.1.1), né ricompense dall'ambiente, come invece accade nel *Reinforced Learning* (vedi Paragrafo 1.1.4). Può sembrare un po' misterioso immaginare cosa possa riuscire a estrapolare il sistema dai dati senza alcun *feedback* dato dall'etichetta o dall'ambiente. Tuttavia, è possibile sviluppare un *framework*[6] per fare *Unsupervised Learning* nel quale l'obiettivo è quello di costruire rappresentazioni dell'*input* in modo che possano risultare utili per prendere decisioni, prevedere *input* futuri e in generale ricercare modelli il più possibile esenti da rumore [16]. Due esempi classici di *Unsupervised Learning* sono il *clustering* e la riduzione della dimensionalità. Il *clustering* è una tecnica che consiste nel raggruppare un insieme di oggetti in modo che quelli dello stesso gruppo (*cluster*) siano più simili (rispetto a una determinata proprietà) tra loro rispetto a quelli di altri *cluster*. In generale esistono molte tecniche per fare *clustering* e la scelta dipende dalle applicazioni e dalle proprietà dei dati. Una delle tecniche di *clustering* più utilizzata è senza dubbio *K-means* [17]. I dati del mondo reale hanno solitamente un'elevata dimensionalità, è necessario quindi ridurne la dimensionalità per riuscire a gestirli in maniera adeguata. La riduzione della dimensionalità è una tecnica di trasformazione dei

[6]Il framework è un insieme di regole, linee guida e metodologie che forniscono una struttura per la costruzione di un sistema, un prodotto o un processo. In informatica, un *framework* può essere utilizzato per creare software, sviluppare applicazioni o semplificare lo sviluppo di soluzioni.

dati ad alta dimensionalità in una rappresentazione a dimensione ridotta. Questa pratica è importante in molti ambiti, poiché attenua la *curse of dimensionality*[7] e altre proprietà indesiderate degli spazi ad alta dimensionalità [19].

1.1.3 Semi-Supervised Learning

Il *Semi-Supervised Learning*, anche chiamato apprendimento semi-supervisionato in italiano, è un tipo di *machine learning* in cui dei dati che vengono passati all'algoritmo solamente una piccola parte è etichettata. L'obiettivo è quello di usare i dati etichettati per riuscire a fornire un'etichetta automaticamente anche ai dati che ne sono privi. Esistono diverse tecniche per fare questo, come ad esempio le tecniche di propagazione delle etichette [20], le tecniche di *co-training* [21], le tecniche di *self-training* [22].

1.1.4 Reinforced Learning

Nel *Reinforced Learning*, anche chiamato apprendimento per rinforzo in italiano, un agente è connesso all'ambiente tramite percezione e azione, come visibile in Figura 5. A ogni step di interazione, l'agente riceve in ingresso, i, un'indicazione dello stato corrente, s, dell'ambiente; quindi l'agente sceglie un'azione, a, per generare un *output*. L'azione cambia lo stato dell'ambiente e il valore di questa transizione di stato viene comunicato all'agente attraverso un segnale di rinforzo, r. Il comportamento dell'agente, B, dovrebbe scegliere azioni che tendono a massimizzare il segnale di rinforzo. Di seguito vengono elencate le diverse fasi dell'apprendimento per rinforzo.

- L'agente inizialmente agisce in maniera casuale, non avendo nessuna conoscenza dell'ambiente;
- interagendo con l'ambiente l'agente inizia a prendere decisioni e ottenere così le ricompense;

[7] Il *curse of dimensionality*, la maledizione della dimensionalità in italiano, è un termine introdotto da Bellman per descrivere il problema causato dall'aumento esponenziale del volume associato all'aggiunta di ulteriori dimensioni allo spazio euclideo [18].

- ottenendo le ricompense l'agente "capisce" come deve agire e adatta la sua *policy* con l'obiettivo di massimizzare la ricompensa.

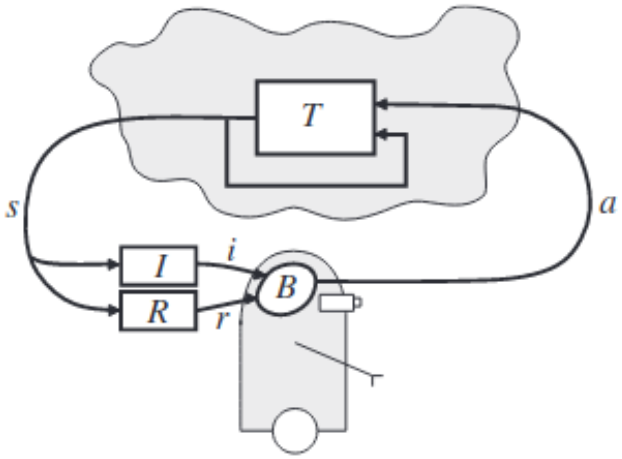


Figura 5: Modello standard di *reinforced learning* (fonte: [23].)

1.1.5 Machine Learning in Bioinformatica

L'apprendimento automatico viene utilizzato in tantissimi ambiti di ricerca [24]: analisi predittiva e processo decisionale intelligente, sicurezza informatica, *internet of things* e città *smart*, previsione del traffico, trasporto pubblico, *healthcare*, pandemia dovuta al COVID-19, NLP ecc. Questa tecnologia trova spazio anche all'interno della bioinformatica, ovvero l'applicazione di tecniche di calcolo per acquisire e interpretare i dati biologici. È un campo interdisciplinare tra informatica, matematica, statistica, biologia e genetica. La crescita esponenziale di dati biologici disponibili ha sollevato due problemi: trovare un modo efficiente per immagazzinare e gestire dati di questo tipo e riuscire a sfruttarli per arrivare a estrarre informazioni utili. I domini in cui vengono utilizzate tecniche di *machine learning* sono molte: in Figura 6 vengono mostrati sei diversi domini nei quali si utilizzano metodi computazionali in ambito bioinformatico: genomica, proteomica, microarray, biologia dei sistemi, evoluzione e *text mining*.

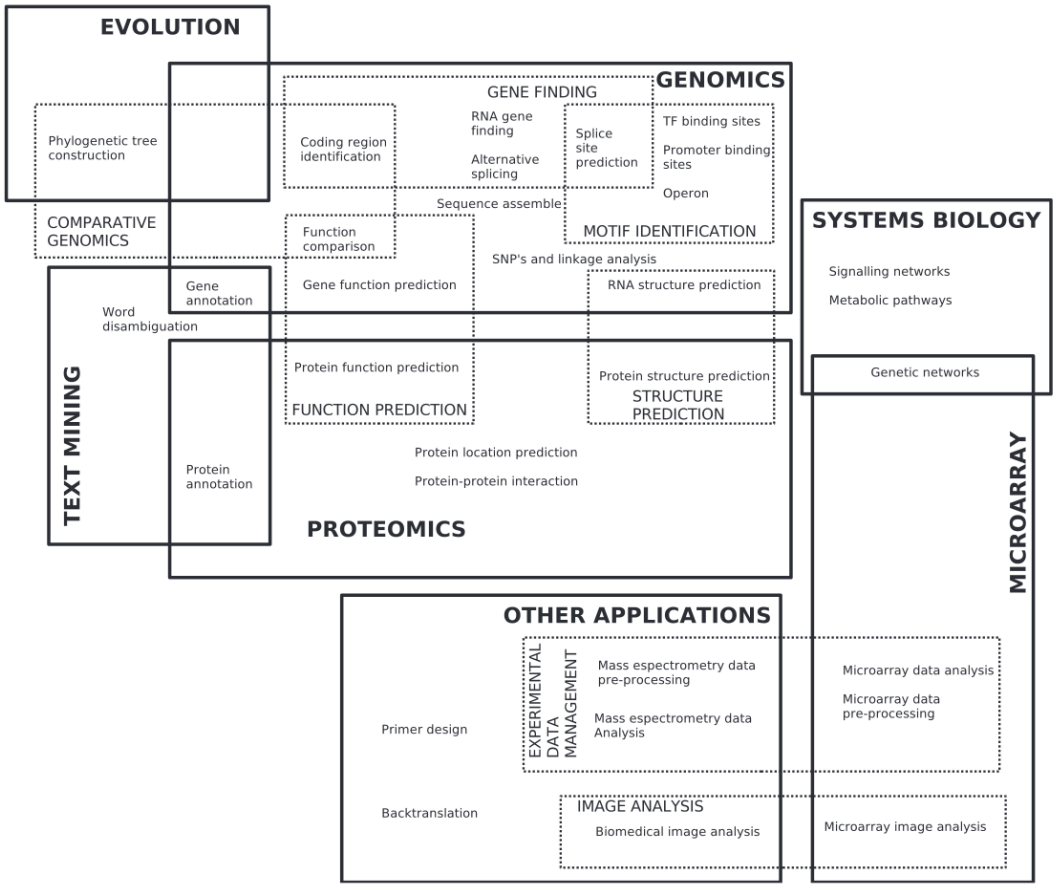


Figura 6: Classificazione dei campi in cui vengono applicati metodi di apprendimento automatico in bioinformatica (fonte: [25]).

La genomica è una delle aree più importanti in bioinformatica. Il numero di sequenze nucleotidiche e proteiche disponibili è in continuo aumento. In Figura 7 è possibile vedere la crescita di dati disponibili dal 1982 al 2020. GenBank è una banca dati a libero accesso e senza restrizione, istituita nel 1982, che riporta tutte le sequenze di nucleotidi e le relative proteine ottenute dopo la loro traduzione. GenBank riceve le proprie informazioni dai risultati ottenuti su oltre 300.000 organismi da laboratori distinti sparsi in tutto il mondo, rappresentando il più importante punto di riferimento nel relativo campo di ricerca. A febbraio 2020, conteneva oltre 216 milioni di loci[8] e oltre 399 miliardi di basi da più di 216 milioni

[8]In genetica, il termine *locus* genico (o più semplicemente *locus*, plurale *loci*) designa la posizione, stabile, di un gene o di un marcatore genico all'interno di un cromosoma.

di sequenze riportate.

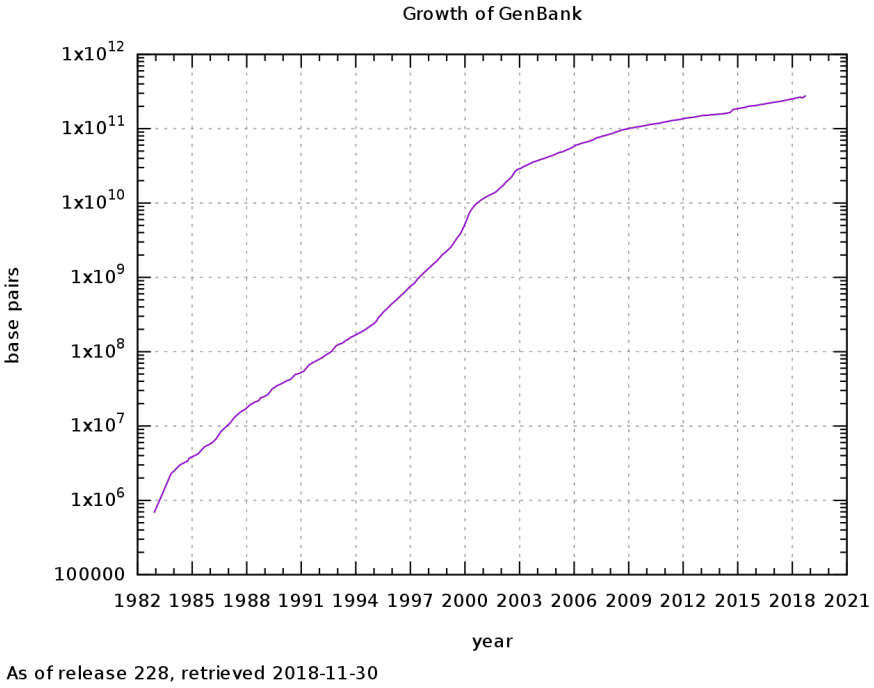


Figura 7: Crescita di GenBank (fonte: [26]).

Tutti questi dati hanno bisogno di essere elaborati in modo da estrapolare informazioni utili. Come prima cosa è possibile estrarre struttura e posizione dei geni dalle sequenze del genoma [27], inoltre è anche possibile identificare gli elementi regolari [28]. Esistono inoltre approcci per estrarre caratteristiche comuni tra gli RNA noti per la previsione di nuovi geni RNA nelle regioni non annotate dai genomi procariotici e arcaici [29].

Nella proteomica i metodi computazionali vengono usati per la previsione della struttura delle proteine. Le proteine hanno una struttura molto complessa composta da migliaia di atomi e legami, di conseguenza il numero di possibili strutture è estremamente elevato. La previsione della struttura risulta quindi un problema combinatorio molto complesso e richiede una buona ottimizzazione.

Attraverso l'applicazione dei metodi computazionali è possibile anche gestire dati sperimentali complessi. Il campo più noto dove troviamo applicazioni di

questo tipo sono i *microarray*[9]. Possono essere utilizzati, ad esempio, per la classificazione di tumori, la previsione della risposta ai farmaci, la scoperta di nuovi geni e biomarcatori. Inoltre i *microarray* possono anche essere utilizzati per l'analisi dell'espressione genica durante lo sviluppo, lo stress o la risposta ai trattamenti farmacologici [30].

Nei paragrafi che seguono sono stati discussi alcuni esempi pratici di applicazione di tecniche di apprendimento automatico in campo bioinformatico e medico.

1.1.6 Apprendimento automatico per la previsione di varianti non codificanti associate a malattie

All'interno di tutte le variazioni genetiche, quelle riguardanti le malattie rappresentano una piccolissima minoranza rispetto all'insieme più ampio di variazioni genomiche non deleterie. Questo comporta uno sbilanciamento tra i sottoinsiemi che compongono le variazioni genomiche. Quello che si vorrebbe fare è usare tecniche di *machine learning* per individuare varianti non codificanti associate alle malattie, ma la scarsità di osservazioni inficia l'efficacia delle tecniche. Lo stato dell'arte dei metodi basati sul *machine learning* non adotta tecniche specifiche che tengano conto dello sbilanciamento dei dati e questo porta inevitabilmente a una riduzione della sensibilità e della precisione nei risultati.

In questo contesto gli algoritmi di apprendimento classici come quelli collegati a *support vector machine* [31] o *artificial neural networks* [32] non riescono a generalizzare in maniera sufficiente poiché di solito prevedono la classe di minoranza con una precisione e sensibilità molto bassa. Nel campo della previsione di varianti genetiche associate a tratti o malattie, questo si riduce a prevedere erroneamente la maggior parte delle varianti associate alla malattia come non associate alla malattia stessa. Viene così limitata in modo significativa l'utilità dei metodi di apprendimento supervisionato per la previsione di nuove varianti non codificanti.

Per affrontare questo problema è stato sviluppato *hyper*SMURF [33] sigla che sta a indicare: *hyper-ensemble of SMOTE Undersampled Random Forest*. Questo

[9]Un *microarray* è una piastra di vetro o policarbonato che contiene migliaia di probe, ovvero sequenze di DNA, genetiche fissate sulla sua superficie, utilizzata per monitorare l'espressione genica su larga scala.

metodo adotta strategie di apprendimento che tengono conto dello sbilanciamento, ovvero tecniche di ricampionamento e su un approccio *hyper-ensemble*: simultaneamente la classe di minoranza viene sovracampionata e la classe di maggioranza viene sottocampionata in modo da generare dati di addestramento bilanciati, ciascuno dei quali verrà poi usato per addestrare un insieme di *Random Forest*. La sua struttura è visibile in Figura 8.

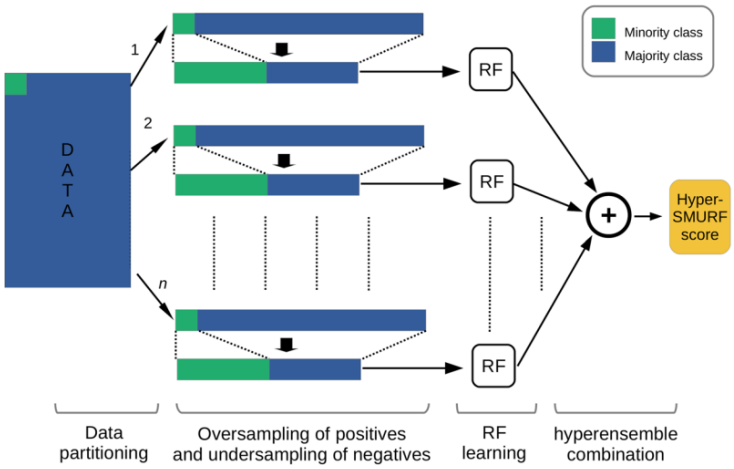


Figura 8: Schema di *hyper*SMURF. Esso divide la classe maggioritaria (rettangoli blu) in n partizioni e per ognuna di esse vengono usate tecniche di sovracampionamento in modo da generare esempi aggiuntivi dalla classe minoritaria (rettangoli verdi). Contemporaneamente un numero simile di esempi viene sottocampionato dalla classe maggioritaria. Successivamente *hyper*SMURF addestra in parallelo n *Random Forest* usando i dati bilanciati e infine combina le predizioni degli n insiemi secondo un approccio *hyper-ensemble* (fonte: [33]).

.

Successivamente le predizioni dei modelli addestrati sono combinate attraverso un approccio chiamato *hyper-ensemble* per ottenere una predizione complessiva "concordata". Vengono addestrati, in parallelo, n *Random Forest*, una per ogni insieme di dati, e vengono combinate le loro previsioni facendo la media delle probabilità, si ottiene un *hyper-ensemble* poiché l'*input* è a sua volta un insieme di *decision tree*.

Il vantaggio risiede nel fatto di avere molta diversità nei dati di addestramento,

inoltre il bilanciamento tra esempi positivi e negativi evita che si abbia una polarizzazione verso la classe maggioritaria. Tutto ciò rende il modello maggiormente in grado di generalizzare, mentre l'approccio di *hyper-ensemble* fornisce apprendimento più accurato e previsioni più robuste. Per testare il modello viene utilizzata una 10-*cross-validation* per assicurarsi che le varianti della stessa malattia non si presentino insieme nel *training set*, o insieme di addestramento, e in quello di *test*, falsando così i risultati. La misura vera e propria è affidata all'AUPRC, ovvero *Area Under the Precision and Recall*, all'AUROC, *Area Under the Receiver Operating Characteristic* e usando la *precision*, *recall* e *F-score*, discussi in Paragrafo 1.1.1.

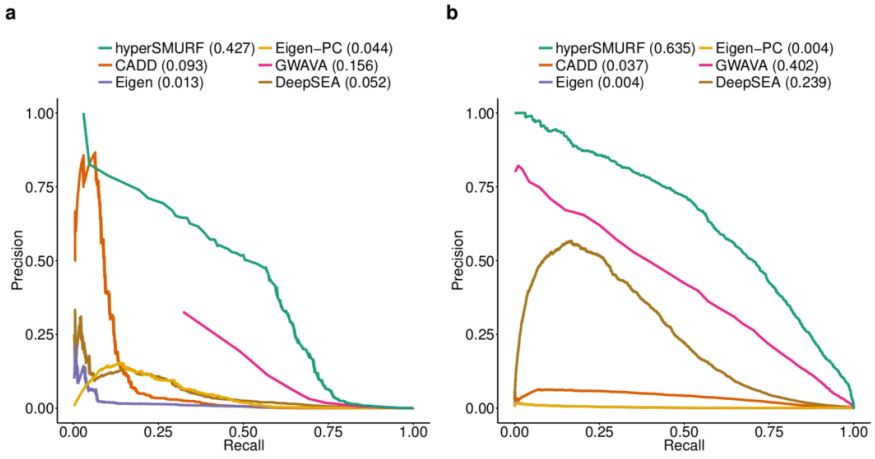


Figura 9: Confronto della curva *precision/recall* tra i vari modelli usando due tipi diversi di dati: a) *Mandelian regulatory mutations*, b) *GWAS regulatory hits*. Il numero tra parentesi rappresenta il valore di AUPRC (fonte: [33]).

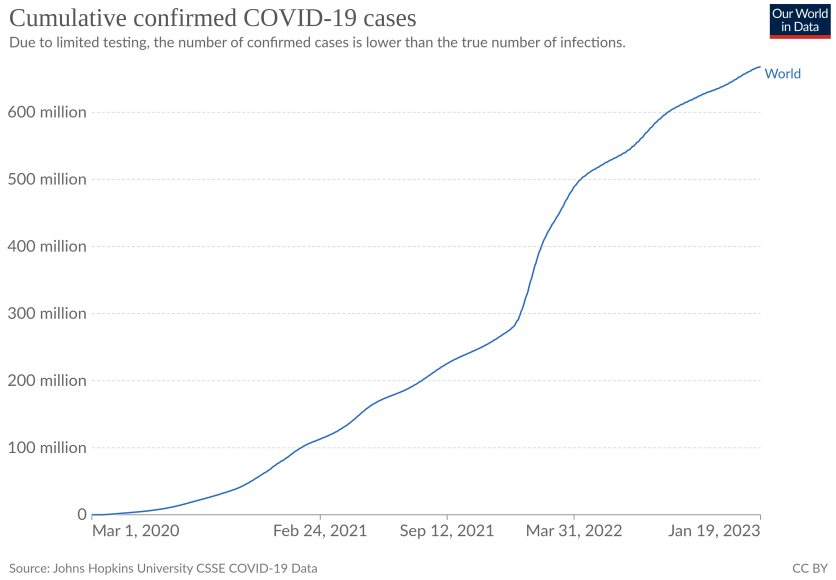
I risultati sono visibili in Figura 9 dove è possibile trovare un confronto tra i vari metodi usando la curva *precision-recall*. Si nota chiaramente come *hyper*SMURF raggiunga migliori risultati rispetto ai metodi classici.

1.1.7 Apprendimento automatico per la previsione del rischio di COVID-19

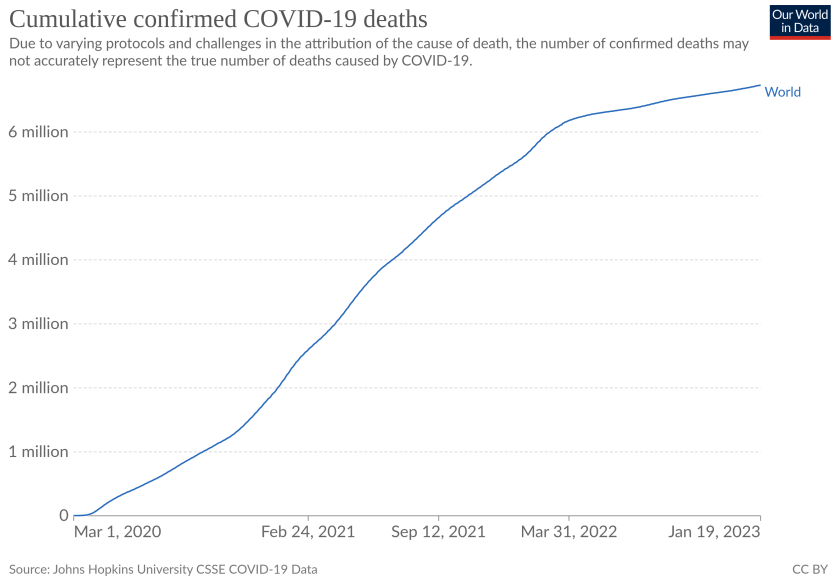
Negli ultimi anni una grave sindrome respiratoria (SARS-Cov-2) ha colpito il mondo provocando una pandemia mondiale e causando 663 milioni di casi accertati e quasi 7 milioni di morti accertati [34]. In Figura 10 sono illustrati i grafici dei casi totali e morti totali da quando è scoppiata la pandemia.

Da quando il virus ha iniziato a diffondersi tra la popolazione, ospedali e pronto soccorso sono stati invasi da pazienti per i quali era necessario sapere se avessero contratto la malattia o meno ma sopratutto la gravità di quest'ultima. Il virus era ed è in grado di causare anomalie nelle radiografie del torace (CXR)[10]. Questo tipo di esame però non basta per determinare la presenza o meno della malattia a causa della sua bassa sensibilità. Il CXR ha però il vantaggio di essere economico e si vuole cercare di sfruttarlo assieme a ulteriori considerazioni e criteri per riuscire a prevedere il rischio di aver contratto il virus con conseguente scoperta della gravità. L'obiettivo è quello di riuscire a creare un sistema che riesca a estrarre le variabili radiologiche, cliniche e di laboratorio più rilevanti che siano in grado di migliorare la previsione del rischio del paziente. Si vogliono inoltre ottenere criteri utilizzabili dai medici nel momento in cui devono decidere il rischio del paziente, inteso come gravità della malattia. In [36] viene illustrato un modello di previsione del rischio per il paziente. Il modello è in grado di selezionare le più importanti variabili cliniche e di laboratorio tenendo anche conto di punteggi radiologici che derivano dalla valutazione del CXR da parte dei radiologi e due punteggi di "coinvolgimento polmonare" calcolati usando alcune delle *deep neural network* più performanti per la diagnosi del rischio da COVID-19 [37–39], come *ResNet* [40, 41], *Inception-Net* [42,43] o *VGG* [44,45]. Successivamente viene eseguita una fase di imputazione dei dati mancanti causata dall'integrazione di più fonti di dati. Le tecniche usate sono state la *Multiple Imputations by Chained Equations* (MICE [46]), sia utilizzando la corrispondenza media predittiva (*micePMM* [47]), sia classificatori *Random Forest* (*miceRF*) come modello di imputazione di *bias*, e *missForest* [48]. In seguito viene

[10] *Chest-X-Ray* è un esame radiologico che utilizza la radiazione a raggi X per creare immagini del torace. Questo esame viene spesso utilizzato per diagnosticare o monitorare una serie di condizioni mediche, tra cui malattie polmonari (come la bronchite, l'enfisema o la polmonite), malattie cardiache (come la cardiopatia), tumori e altri problemi che interessano la zona toracica.



(a)



(b)

Figura 10: Grafico che mostra a) casi totali e b) morti totali nel mondo da quando è scoppiata la pandemia a oggi (fonte [35]).

costruito un algoritmo robusto che combina Boruta [49, 50] con metodi di *feature selection*, basati sulle permutazioni e incorporati nelle *random forest* [51].

	model	AUC (var)	Sensitivity (var)	Specificity	F1-score	Accuracy
missForest	RF	**0.81** (0.00007)	**0.72** (0.00016)	0.76 (0.00006)	**0.62** (0.00009)	**0.74** (0.00006)
	AT	0.67 (0.00013)	0.51 (0.00039)	0.83 (0.00020)	0.53 (0.00028)	0.67 (0.00013)
	GLM	0.80 (0.00001)	0.56 (0.00002)	0.86 (0.00001)	0.62 (0.00002)	0.71 (0.00001)
miceRF	RF	0.79 (0.00011)	0.70 (0.00034)	0.74 (0.00012)	0.60 (0.0002)	0.72 (0.00014)
	AT	0.65 (0.00027)	0.48 (0.00079)	0.82 (0.00022)	0.50 (0.00062)	0.65 (0.00027)
	GLM	0.78 (0.0005)	0.53 (0.00025)	0.85 (0.00004)	0.59 (0.00014)	0.69 (0.00009)

Figura 11: Tabella con le misure di *performance* calcolate usando *random forest* (RF), alberi associativi derivati (AT), modelli generalizzati (GLM) (fonte [36]).

Le *feature* selezionate sono usate come *input* per *random forest*, (vedi Paragrafo 1.1.1) e per alberi associativi derivati [52]. A differenza dei *random forest* che producono un gran numero di regole di difficile comprensione, gli alberi associativi derivati sono costruiti partendo dai primi ma producendo regole più semplici che possono essere facilmente valutate e interpretate dai medici. I risultati ottenuti dai due algoritmi sono comparati a quelli ottenuti dai modelli generalizzati (GLM [53]). In Figura 11 è possibile vedere i punteggi ottenuti con i modelli citati precedentemente.

Capitolo 2

Dataset

Un insieme di dati, *dataset* in inglese, è una raccolta di dati in cui ogni riga rappresenta un'osservazione, o istanza, e ogni colonna costituisce un attributo, o caratteristica, dell'istanza. I metodi di apprendimento hanno bisogno di dati per poter addestrare modelli. L'obiettivo è riuscire a creare modelli che siano in grado di generalizzare ed effettuare previsioni affidabili sui nuovi dati. I dati possono influire significativamente sulle prestazioni delle predizioni, pertanto è importante utilizzare un insieme di dati ben equilibrato, rappresentativo e qualitativo.

In questo lavoro è stato usato apprendimento supervisionato quindi l'insieme di dati aveva anche un'etichetta. L'obiettivo è stato addestrare modelli in modo che riuscissero a estrapolare la relazione che sussiste tra i dati, X, e l'etichetta, y, per far sì che, fornendo in *input* nuove istanze, restituissero l'etichetta predetta sulla base dei dati passati precedentemente. In questo capitolo sono stati spiegati i dati multi-omici, ovvero la tipologia di dati usata durante gli esperimenti, la fonte utilizzata per reperire i dati e approfonditi i dati impiegati. Successivamente sono state illustrate le varie fasi del preprocessing ed è stata fatta una rassegna di tutte le tecniche di di *dimensionality reduction* usate durante gli esperimenti.

2.1 Dati Multi-Omici

I dati multi-omici sono un insieme che contiene le variazioni molecolari su più livelli quali: genomica, epigenomica, trascrittomica, proteomica, metabolomica

e microbiotica [54]. La disponibilità di questo tipo di dati ha completamente rivoluzionato il campo della medicina e della biologia. In Figura 12 è possibile vedere i livelli spiegati precedentemente, di seguito sono analizzati nel dettaglio.

- La genomica è il campo che si occupa dell'identificazione dei geni e delle varianti geniche associate a una malattia o in risposta a determinati medicinali.

- L'epigenomica è il campo dell'identificazione di modificatori di DNA o delle proteine associate al DNA. Le modifiche epigenetiche del genoma possono anche agire come marcatori per sindromi metaboliche, malattie cardiovascolari e disturbi fisiologici. Queste modifiche possono essere specifiche per cellula e tessuto: è quindi fondamentale identificare le modifiche epigenetiche durante gli stati nativi della malattia.

- La trascrittomica è il campo che studia le proprietà associate alla produzione di RNA dal DNA. L'obiettivo di questa area di ricerca è la comprensione di come il DNA venga trascritto in RNA. Sebbene solo il 2% del DNA venga tradotto in proteine, quasi l'80% del genoma viene trascritto e tale processo include RNA codificante, RNA corto [55], microRNA [56], *piwi* RNA [57] e piccoli RNA nucleari [58]. Lo studio della trascrittomica può aiutare a identificare molti problemi di salute, tra cui: cancro [59], malattie neurodegenerative (come l'Alzheimer [60]), infiammatorie [61] e metaboliche [62].

- La proteomica è il campo nel quale si cerca di identificare gli strati, le modifiche e le interazioni delle proteine a livello del genoma. Queste modifiche sono coinvolte nella manutenzione della struttura e della funzione cellulare.

- Il metaboloma è l'insieme di tutti i metaboliti presenti in una cellula, tessuto o organismo, compresi piccoli molecolari, carboidrati, peptidi, lipidi, nucleotidi e i prodotti catabolici. Rappresenta il prodotto finale della trascrizione genica e consiste sia in molecole di segnalazione sia strutturali. La dimensione del metaboloma è molto più piccola rispetto alla dimensione del proteoma e quindi è più facile da studiare.

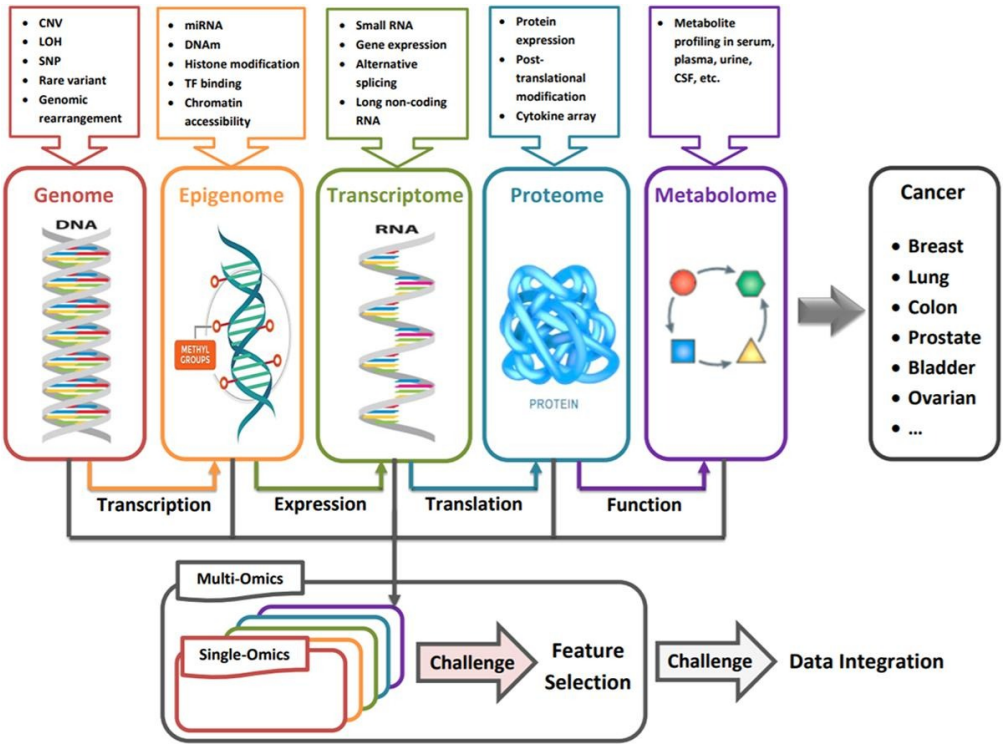


Figura 12: Diagramma delle relazioni tra i vari tipi di dati omici (fonte: [64]).

- La microbiomica consiste in tutti i microorganismi di una comunità. I microbi sono ovunque: sono stati trovati sulla pelle umana, sulle superfici mucose e nell'intestino. Il microbioma presente negli esseri umani, oltre che molto diffuso, è molto complesso, ad esempio nell'intestino troviamo circa 100 trilioni di batteri [63]. La microbiota è stata trovata coinvolta in diabete, obesità, cancro, colite, malattie cardiache e autismo. Pertanto la caratterizzazione del microbioma di un organismo è di grande interesse medico.

Grazie alla ricerca ci sono stati importanti progressi in diversi campi omici e si è capito che la risposta a una domanda in medicina non è da ricercare in un solo tipo di dato poiché questi dati si influenzano tra di loro: per esempio il microbioma influenza l'espressione genica e proteica che a sua volta influenza il metabloma. È quindi necessario studiare i dati nella loro interezza sfruttando i dati multi-omici per comprendere lo stato nativo e alterato di un organismo attraverso l'analisi dei dati provenienti da diverse fonti omiche.

Esistono diversi database che forniscono dati multi-omici di pazienti, uno di questi è il *The Cancer Genome Atlas*.

2.1.1 The Cancer Genome Atlas (TCGA)

Il *Cancer Genome Atlas* [65] è il programma scientifico di riferimento per la genomica del cancro e ha favorito la caratterizzazione sistematica di diverse alterazioni genomiche alla base dei tumori umani. Attualmente ha caratterizzato oltre 11000 tumori di 33 tipi di cancro. Si tratta di uno sforzo congiunto tra *Nation Cancer Institute* e il *National Human Genome Research Institute* e riunisce ricercatori di diverse discipline e istituzioni di ogni parte del mondo.

I dati forniti sono composti da 627 pazienti e di ognuno sono presenti i dati relativi a proteine, CNV, mRNA, miRNA e la relativa etichetta (che indica la presenza o assenza di evento tumorale). I primi 3 tipi di dati sono stati presi da TCGA tramite il *package curatedTCGAData*[1] da Bioconductor[2]. Essi sono dati sul carcinoma mammario invasivo (TCGA-BRCA). Questo è un tipo di cancro che si sviluppa all'interno della mammella ed è in grado di diffondersi alle altri parti del corpo attraverso linfonodi o vasi sanguigni. I dati sono stati ottenuti facendo riferimento alla versione del genoma umano hg19: questa versione è stata pubblicata nel 2009 dal progetto internazionale del genoma umano (IGHP)[3] e contiene informazioni su sequenza del DNA umano, sequenze dei geni, i siti di regolazione genica e le regioni non codificanti del DNA. Per quanto riguarda invece le etichette sono state prese da un *dataset* noto come TCGA-CDR [66], curato manualmente per avere dati clinici e di sopravvivenza il più affidabili possibile. Nei paragrafi che seguono sono stati approfonditi tali dati.

Proteine Le proteine sono le macromolecole che svolgono essenzialmente ogni compito all'interno della cellula. Questi dati rappresentano i loro livelli di espressione. Tra i loro compiti, quelli più importanti sono:

[1] https://bioconductor.org/packages/release/data/experiment/html/curatedTCGAData.html

[2] La missione del progetto Bioconductor è quella di sviluppare, supportare e diffondere software open source gratuito che faciliti un'analisi rigorosa e riproducibile dei dati provenienti da saggi biologici attuali ed emergenti.

[3] https://www.genome.gov/human-genome-project

- strutturali: alcune proteine formano la struttura delle membrane cellulari e altre sostengono la forma della cellula,
- catalitici: molte proteine sono enzimi che catalizzano reazioni chimiche all'interno della cellula,
- regolatori: alcune proteine regolano l'espressione genica e la risposta allo stress,
- di trasporto: alcune proteine fungono da trasportatori di ioni e molecole attraverso la membrana cellulare,
- immunitari: alcune proteine fungono da anticorpi che proteggono la cellula da agenti esterni,
- comunicativi: alcune proteine fungono da messaggeri chimici che comunicano tra le cellule.

I dati utilizzati durante gli esperimenti sono stati ottenuti con una tecnica nota come RPPA [67] (*Reverse Phase Protein Array*).

mRNA RNA messaggero (noto con l'abbreviazione di mRNA o con il termine più generico di trascritto) è una delle principali molecole in grado di trasportare l'informazione genetica dal nucleo della cellula, dove si trova il DNA, al citoplasma, dove avviene la sintesi proteica. I dati indicano i suoi livelli di espressione. Essi sono ottenuti tramite una tecnica chiamata *RNA-sequencing* [68].

miRNA microRNA, noto con l'abbreviazione di miRNA, sono piccoli RNA (non codificanti per proteine) che svolgono funzioni di regolazione all'interno della cellula. I dati indicano i suoi livelli di espressione. Essi sono ottenuti, come per mRNA, tramite *RNA-sequencing*.

CNV *Copy Number Variants* (CNV), o varianti nel numero di copie, sono le varianti genetiche, tra cui inserzioni, delezioni e duplicazioni di segmenti di DNA. I dati si riferiscono al tratto genetico che coinvolge il numero di copie di un particolare gene presente nel genoma di un individuo. Per ogni gene viene indicato se lo

stesso abbia subito delezione (come la perdita di una o più copie), amplificazione (acquisizione di una o più copie), neutro (nessuna modifica). I dati possono può assumere i seguenti valori: 0 (neutro), 1 o 2 (amplificazioni), -1 o -2 (delezioni).

Etichetta Rappresentano la y descritta nel Paragrafo 1.1.1. In Figura 13 è possibile vedere un diagramma a barre che mostra come i dati sono distribuiti tra sani e malati. Le etichette fanno riferimento a una misura nota come PFI (*Progression Free Interval*) dove:

- 1 indica che il paziente ha un nuovo evento tumorale, che sia una progressione della malattia, una recidiva locale, una metastasi a distanza, nuovi tumori primari in tutti i siti o sia morto con il cancro senza nuovo evento tumorale, compresi i casi con un nuovo evento tumorale il cui tipo è N/A,
- 0 indica che il paziente non ha avuto alcun evento tumorale.

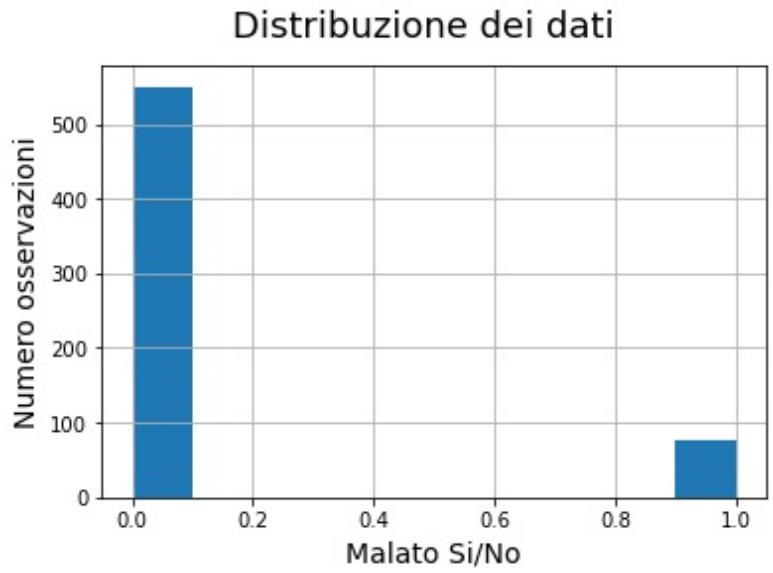


Figura 13: Istogramma che mostra quanti pazienti risultano malati in base a una misura nota come PFI (*Progression Free Interval*) in cui: 1 sta a indicare la presenza di un evento tumorale in un paziente (colonna di destra); 0 l'assenza di un evento di questo tipo (colonna di sinistra).

2.2 Preprocessing

La prima fase del lavoro ha riguardato la manipolazione dei dati per renderli di più facile analisi e interpretazione per l'utilizzo dell'algoritmo che verrà applicato successivamente. Questo processo è importante poiché i dati grezzi spesso non sono adatti per l'algoritmo scelto e questo può influire sulle prestazioni finali. Di seguito sono illustrati i diversi passaggi del *preprocessing*.

- Codifica: processo di gestione delle variabili categoriche. Ciò consiste nel trasformare queste variabili non numeriche in una rappresentazione numerica utilizzabile dai modelli di *machine learning*.

- Normalizzazione dei dati: consiste nel trasformare i valori di una o più variabili usando una scala comune. Questo viene fatto poiché all'interno del *dataset* potrebbero esserci variabili che usano scale diverse. Prendiamo per esempio un dataset con all'interno due variabili numeriche, "altezza" e "peso": ovviamente non usano la stessa scala, la prima si misura in centimetri e la seconda in kg.

- Pulizia dei dati: consiste nella rimozione di valori duplicati, valori completamente fuori scala, detti *outlier* e valori mancanti o nulli. Questi ultimi possono anche essere imputati. L'imputazione è il processo di sostituzione di questo tipo di dati con valori sintetici. Questa pratica viene utilizzata poiché molti algoritmi di *machine learning* non possono gestire valori mancanti e il loro effetto sulle prestazioni dei modelli può essere significativo. Esistono diverse tecniche per fare imputazione di dati. Alcuni esempi sono: *mean imputation*, *median imputation*, *regression imputation*, *k-nearest neighbors (KNN)*, *multiple imputation* ([69, 70]).

- *Dimensionality reduction*: consiste nella riduzione del numero di caratteristiche di un *dataset* ad alta dimensionalità in uno a più bassa dimensionalità pur mantenendo il più possibile l'informatività dei dati. Questo processo consente di ridurre la complessità del modello, migliorare la sua capacità di generalizzazione e la precisione delle previsioni. La riduzione della dimensionalità può essere fatta in diversi modi, usando tecniche di *feature selection*,

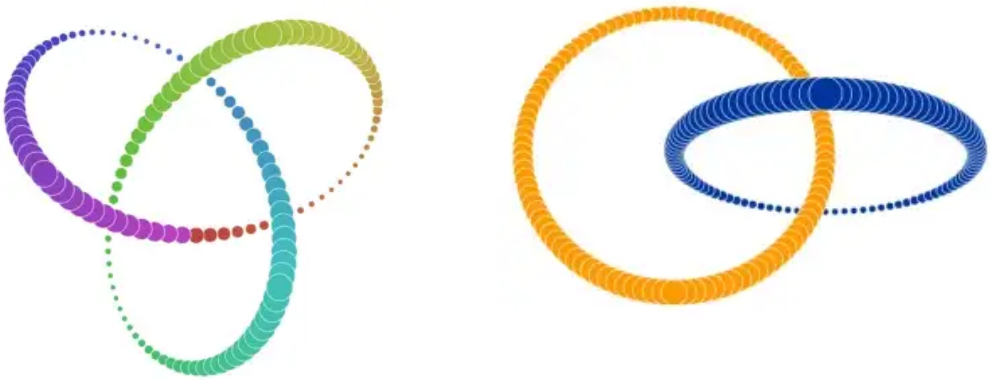
	Proteine	mRNA	miRNA	CNV
numero di *feature*	216	20501	1046	24776

Tabella 1: Tabella che illustra il numero di *feature* per ogni *dataset* fornito.

feature extraction o tecniche come t-SNE, algoritmo di visualizzazione non lineare utilizzato per ridurre la dimensione di un *dataset* in uno spazio a bassa dimensione, pur mantenendo le relazioni di prossimità tra i punti. Il primo modo consiste nella selezione delle caratteristiche; ovvero nella scelta delle variabili, o *feature* appunto, più rilevanti per il modello di *machine learning*, al fine di ridurre la complessità del modello, migliorare la sua capacità di generalizzazione e accelerarne la convergenza durante l'addestramento. Il secondo modo consiste nel costruire nuove *feature* in funzione di quelle di partenza.

2.3 Riduzione della dimensionalità

Negli ultimi anni i dati disponibili per applicazioni di *machine learning* in ambiti come *mining* di testo, *computer vision* e biomedico stanno aumentando esponenzialmente sia in termini di campioni sia in termini di numero di dimensioni. L'enorme numero di *feature* dei *dataset* attualmente disponibili porta diversi svantaggi: rallentamento significativo degli algoritmi di *learning*, peggioramento della *performance* dei suddetti algoritmi e difficoltà nell'interpretazione del modello. Molto spesso i *dataset* che vengono utilizzati per il *machine learning* hanno un'elevata dimensionalità, basti pensare che i *dataset* usati negli esperimenti hanno 627 osservazioni (ovvero i pazienti) e un numero di *feature* molto elevato (vedi Tabella 1). Per gestire adeguatamente questi dati del mondo reale è necessario ridurne la dimensionalità. La riduzione della dimensionalità è la trasformazione di dati ad alta dimensione in una rappresentazione significativa di dimensionalità ridotta [71]. Idealmente, la rappresentazione ridotta dovrebbe avere una dimensionalità che corrisponde alla dimensionalità intrinseca dei dati (vedi Paragrafo 2.3.4). Esistono diversi modi per eseguire riduzione della dimensionalità: in questo lavoro sono state usate tecniche di *feature selection*, *feature extraction*. La riduzione della



(a) (b)

Figura 14: Esempi di dati non separabili linearmente (fonte: [76] CC-BY 2.0).

dimensionalità è importante in molti domini, poiché attenua la *curse of dimensionality* [18] e altre proprietà desiderate degli spazi ad alta dimensionalità [19]. La riduzione della dimensionalità facilita la classificazione, la visualizzazione e la compressione di dati ad alta dimensionalità. Esistono diverse tecniche lineari come *Principal Component Analysis* (PCA) [72], *Linear Discriminant Analysis* (LDA) [73] e *Singular Value Decomposition* (SVD) [74]. Tuttavia queste tecniche lineari non sono in grado di gestire adeguatamente dati complessi e non lineari. Esistono tecniche di riduzione della dimensionalità non lineari che vengono utilizzate in molti casi in cui i dati hanno una struttura non lineare e non possono essere adeguatamente rappresentati mediante tecniche lineari di riduzione della dimensionalità, i più noti sono UMAP (*Uniform Manifold Approximation and Projection*) (*feature extraction*) e t-SNE (*t-distributed Stochastic Neighbor Embedding*).

2.3.1 t-SNE: t-distributed Stochastic Neighbor Embedding

t-SNE è una tecnica di riduzione della dimensionalità non lineare introdotta in Van der Maaten et al. [75]. Essendo una tecnica non lineare è in grado di separare dati che non possono essere separati da una linea retta. La Figura 14 ne riporta un esempio.

Di seguito sono elencate le diverse fasi dell'algoritmo.

- Inizialmente viene creata una distribuzione di probabilità che rappresenta la similarità tra i vicini. La similarità tra due punti x_i e x_j è la probabilità condizionale $P_{j|i}$ che x_i sceglierebbe x_j come suo vicino se i vicini venissero scelti in base alla funzione di densità di probabilità sotto una gaussiana centrata in x_i [75]. Successivamente viene calcolata la distanza euclidea per ogni punto appartenente al *dataset*. Dopo un processo di normalizzazione, quello che si ottiene è un insieme di valori di probabilità $P_{i,j}$ che sono proporzionali ai valori di similarità. La distribuzione può essere manipolata usando quella che viene chiamata "perplessità", la quale è, più o meno, un numero target di vicini per il nostro punto centrale. In pratica, più alta è la perplessità, più alto è il valore della varianza, ovvero quanto è ampia la curva.

- Successivamente viene ripetuto il processo descritto precedentemente ma questa volta al posto di una distribuzione gaussiana viene usata una t-Student. Questa nuova distribuzione ci fornisce un nuovo insieme di probabilità Q_{ij} ma in uno spazio ridotto. Come visibile in Figura 15, la distribuzione t-Student ha code più lunghe rispetto a una distribuzione gaussiana e ciò consente di visualizzare più facilmente le distanze tra i punti.

- Infine si procede a calcolare le probabilità Q_{ij} nello spazio ridotto in modo tale che esse riflettano nella maniera più precisa possibile le probabilità P_{ij} relative allo spazio iniziale a dimensioni più elevate. Questo procedimento viene svolto misurando la differenza tra le due distribuzioni di probabilità usando la divergenza di Kullback-Liebler [77] che è in grado di confrontare i valori di P_{ij} e Q_{ij}. Per minimizzare la divergenza tra le due dimensioni viene utilizzata la *gradient descent* [75, 78].

2.3.2 Feature Extraction

La *feature extraction* è una tecnica di riduzione della dimensionalità che consiste nella creazione di nuove *feature* in funzione delle variabili di partenza. L'obiettivo è sempre quello di migliorare le prestazioni di apprendimento automatico o, in generale, semplificare il modello. Tra le tecniche di *feature extraction* le più

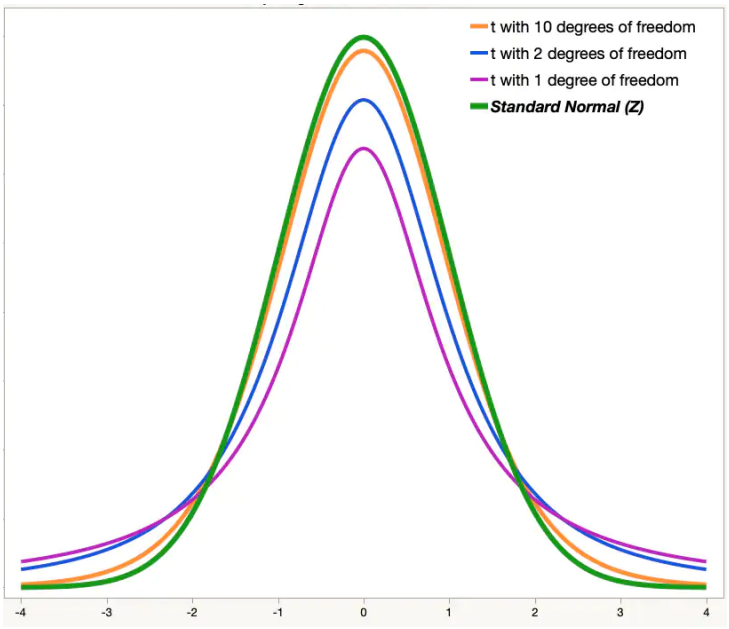


Figura 15: Grafici della densità delle distribuzioni gaussiane (in verde) e t-Student (in arancione, blu e viola) a seconda dei gradi di libertà ovvero al numero di osservazioni indipendenti all'interno del *dataset* di partenza (fonte [79]).

note sono: *Principal Component Analysis*(PCA) [80], *Linear Discriminant Analysis* (LDA) [73], *Singular Value Decomposition* (SVD) [81] e UMAP, illustrato nel Paragrafo seguente.

UMAP: Uniform Manifold Approximation

UMAP [82] è un algoritmo di riduzione della dimensionalità non lineare il cui scopo è rappresentare dati ad alta dimensionalità in uno spazio a bassa dimensione cercando di preservare le relazioni topologiche e spaziali tra i punti dei dati. Centrale è il concetto di *manifold*: chiamata varietà in italiano, si tratta di uno spazio topologico che localmente è simile a uno spazio topologico ben conosciuto (ad esempio lo spazio euclideo n-dimensionale), ma che globalmente può avere proprietà geometriche differenti (ad esempio può essere curvo contrariamente allo spazio euclideo). Per esempio, una superficie sferica è un *manifold* di dimensione 2 in uno spazio tridimensionale. Una curva può essere vista come un *manifold* di dimensione 1 in uno spazio a due dimensioni. La Figura 16 riporta un esempio

di questo fatto, considerando la superficie terrestre. In termini di analisi dei dati,

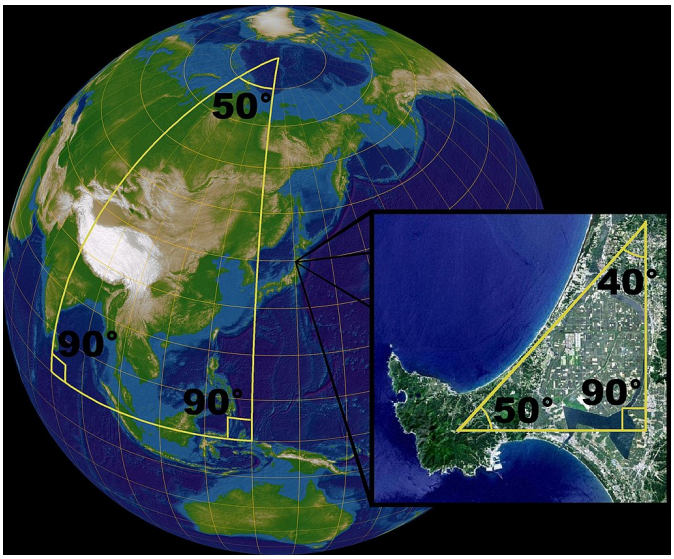


Figura 16: Localmente la superficie terrestre somiglia a un piano, e per questo è una varietà di dimensione 2. Tuttavia tale somiglianza non conserva la distanza tra i punti, in quanto la sfera ha una curvatura diversa. La curvatura incide sulla somma degli angoli interni di un triangolo: nel piano tale somma è sempre 180°, mentre su una sfera è sempre maggiore. Ad esempio, la somma degli angoli interni del triangolo in figura è 230°. La figura in basso a destra è un triangolo in senso euclideo ma non rispetto alla geometria della sfera, in quanto i suoi lati non rappresentano delle geodetiche della sfera (fonte: [83] Dual-licensed under the GFDL and CC-By-SA).

UMAP considera che molti *dataset* ad alta dimensionalità siano distribuiti su un *manifold* sottostante di dimensione inferiore. L'idea è che le proprietà dei dati siano meglio descrivibili da questo *manifold* sottostante piuttosto che dallo spazio completo ad alta dimensionalità.

L'algoritmo UMAP funziona attraverso due fasi principali:

- approssimazione uniforme delle strutture di *manifold*: UMAP utilizza una combinazione di tecniche di apprendimento non supervisionato, come la costruzione di un grafo di vicinato e l'utilizzo di una funzione di costo, per approssimare la struttura dei dati ad alta dimensionalità;

- proiezione su uno spazio a bassa dimensione: dopo aver approssimato la struttura dei dati, UMAP utilizza una tecnica di proiezione non lineare

per rappresentare i dati in uno spazio a bassa dimensione, mantenendo le relazioni topologiche e spaziali.

UMAP utilizza un metodo di costruzione e incorporazione di grafi non lineari per ottimizzare un obiettivo che consente un compromesso tra l'enfatizzazione delle strutture locali e la conservazione delle distanze a livello globale. Questo compromesso è controllato principalmente dai parametri *n_neighbors* e *min_dist* di UMAP. Il parametro *n_neighbors* controlla il numero di vicini di cui viene preservata la topologia locale, le distanze globali vengono preservate quando è alto. Il parametro *min_dist* controlla la distanza minima tra i campioni nell'incorporazione e ciò influisce sulla diffusione dei *cluster*. Valori bassi di *min_dist* consentono a UMAP di enfatizzare la somiglianza di gruppi densi di campioni, mentre valori più grandi si concentrano sulla conservazione di distanze più ampie.

2.3.3 Feature selection

Nel *machine learning*, e in statistica, con il termine *feature selection* si intende il processo di selezione di un sottoinsieme di *feature*, chiamate anche caratteristiche, dimensioni o variabili, rimuovendo quelle irrilevanti, ridondanti o che producono solo rumore [84]. Questa pratica, di solito, porta a una migliore capacità di addestramento, accuratezza più elevata, minore costo computazionale e aumento dell'interpretabilità del modello. La *feature selection* aiuta anche a non incappare nel *curse of dimensionality*.

Esistono diverse tecniche per effettuare *feature selection* le quali possono essere di tipo univariato o multivariato. La differenza tra queste tecniche risiede in come vengono valutate le *feature*. Le tecniche univariate valutano l'importanza di ogni *feature* in modo indipendente dalle altre. Queste tecniche sono indubbiamente più facili da implementare e comprendere rispetto a quelle multivariate ma potrebbero ignorare completamente le relazioni tra le variabili e selezionare *feature* inutili. Le tecniche multivariate valutano l'importanza di una *feature* confrontandola con tutte le altre *feature*. Al contrario delle tecniche univariate, le tecniche multivariate sono più adatte per dataset con molte *feature* ma possono essere più complesse da implementare e comprendere rispetto alle tecniche univariate [85]. Infine le

tecniche multivariate risultano molto più costose a livello computazionale. Di seguito vengono approfondite le tecniche usate durante gli esperimenti.

L'indice di correlazione di Pearson

L'indice di correlazione di Pearson è un metodo che misura la correlazione lineare tra due variabili, spesso indicato con ρ [86]. Viene calcolato come il rapporto tra la covarianza di due variabili e il prodotto delle loro deviazioni standard ed è sempre compreso tra -1 e 1. Data una coppia di variabili casuali (X, Y) è possibile calcolare ρ con la formula seguente:

$$r_{xy} = \frac{\text{cov}(X, Y)}{\sigma_x \cdot \sigma_y}, \tag{4}$$

dove:

- r_{xy} è l'indice di correlazione di Pearson tra le variabili X e Y,
- $\text{cov}(X, Y)$ è la covarianza tra le variabili X e Y,
- σ_x e σ_y sono la deviazione standard delle variabili X e Y.

Parlando dell'indice di correlazione di Pearson è bene definire due concetti: *cut-off* e *p-value*. Il primo viene utilizzato per stabilire se esiste o meno una correlazione significativa tra due variabili. Il *cut-off* viene solitamente stabilito in base al livello di significatività desiderato e può variare tra i diversi studi. Un valore di *cut-off* comune è 0.8, il che significa che una correlazione con un valore di indice di Pearson superiore a 0.8 è considerata significativa e indica una correlazione lineare forte tra due variabili. In questo caso potrebbe essere opportuno scartare una delle due variabili, poiché potrebbe essere considerata ridondante. Tuttavia, la scelta dipende dal contesto specifico del problema e dalla comprensione che si ha delle relazioni tra le variabili.

Il secondo, invece, indica la probabilità che la correlazione osservata tra le due variabili sia dovuta al caso. Un *p-value* basso, solitamente inferiore a 0.05, indica che la correlazione è significativa e che non è probabile che sia stata generata casualmente. Al contrario, un *p-value* elevato indica che la correlazione potrebbe

essere dovuta al caso e che non c'è sufficiente evidenza per supportare l'esistenza di una correlazione significativa.

In altre parole, il *p-value* fornisce una misura del livello di significatività statistica della correlazione tra le variabili. Se il *p-value* è inferiore a un certo livello di significatività, si può rifiutare l'ipotesi nulla e concludere che esiste una correlazione significativa tra le variabili. L'ipotesi nulla, spesso indicata con H_0, è un'affermazione che viene sfidata per essere confermata o smentita tramite i dati a disposizione. L'ipotesi nulla viene accettata finché non è evidente che valga l'ipotesi opposta, ovvero quella alternativa, indicata con H_1. Ovvero:

- ipotesi nulla $H_0 : P(X \geq Y) = P(Y \geq X)$,
- ipotesi alternativa $H_1 : P(X \geq Y) \neq P(Y \geq X)$.

L'ipotesi nulla viene rifiutata quando il *p-value* è minore di un determinato livello di significatività prefissato (tipicamente 0.05). Il rifiuto dell'ipotesi comporta che ci sia evidenza tale che i dati non riescono a fornire abbastanza prove a favore dell'ipotesi nulla.

È bene notare che il *p-value* e il *cut-off* sono entrambi utilizzati per stabilire se una correlazione sia significativa o meno, ma il *p-value* fornisce una misura quantitativa della probabilità che i risultati ottenuti siano casuali, mentre il *cut-off* è un valore soglia fissato arbitrariamente.

Il coefficiente di Spearman

Il coefficiente di correlazione di rango Spearman è una misura non parametrica della monotonicità della correlazione tra due variabili ordinali o tra due variabili quantitative con una distribuzione non-normale. Il coefficiente di correlazione di Spearman è simile al coefficiente di correlazione di Pearson, ma è più adatto per i dati non-normali o ordinali, poiché non richiede che le variabili siano normalmente distribuite. I valori variano tra -1 e $+1$ con 0 che implica una correlazione debole o assente. Le correlazioni con valori -1 implicano una correlazione negativa forte mentre quelle con valori $+1$ implicano una correlazione positiva forte [87]. Anche in questo caso, come con l'indice di Pearson, potrebbe essere opportuno scartare una delle due variabili altamente correlate, poiché potrebbe essere ridondante.

Nella correlazione di Spearman il *p-value* viene utilizzato per determinare se la correlazione tra le due variabili sia significativa o meno. Se il *p-value* è inferiore a un livello di significatività prestabilito (nel mio lavoro ho usato 0.05) si può dire che la correlazione è significativa. Al contrario, se il *p-value* è maggiore di un certo livello di significatività, non c'è evidenza sufficiente per affermare che la correlazione sia significativa. In sintesi, l'indice di correlazione di Spearman fornisce informazioni quantitative sulla forza della correlazione tra le variabili, mentre il *p-value* fornisce informazioni sul significato statistico della correlazione. Il coefficiente viene così calcolato:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}, \tag{5}$$

dove:

- ρ è il coefficiente di correlazione di Spearman,
- n è il numero di osservazioni,
- d_i è la differenza tra la posizione della i-esima osservazione nella prima variabile e nella seconda variabile, ordinate in modo crescente.

Il test di Mann-Whitney

Il test U di Mann-Whitney, anche noto con il nome di test della somma dei ranghi di Wilcoxon o test U di Mann-Whitney, è un test non parametrico dell'ipotesi nulla. Questo test viene spesso utilizzato per determinare se esiste una differenza significativa tra le medie di due gruppi, indipendentemente dalla loro distribuzione di probabilità. Una formulazione generale del test U consiste nell'assumere che:

- tutte le osservazioni provenienti da due gruppi siano indipendenti l'uno dall'altro,
- prese due osservazioni, si possa dire quale sia la maggiore. I valori dovrebbero essere ordinali,
- sotto l'ipotesi nulla H_0, le distribuzioni delle due popolazioni siano identiche,
- l'ipotesi alternativa H_1 dice che le distribuzioni non siano identiche.

Secondo le assunzioni fatte precedentemente, il test è coerente quando: per valori estratti uniformemente a caso X e Y, si verifica se la probabilità che X sia maggiore di Y sia uguale alla probabilità che Y sia maggiore di X.

Il test viene usato per selezionare quindi le *feature* che hanno un valore di *p-value*, rispetto all'etichetta per cui si vuole classificare, minore di una certa soglia (nel mio caso 0.05). Tutte le *feature*, invece, che hanno un valore di *p-value* superiore vengono scartate.

Minimum Redundancy Maximum Relevance: mRMR

Proposto da Peng at al. [88] la *Maximum Relevance and Minimum Redundancy* (mRMR) è una tecnica di *feature selection* multivariata che utilizza l'informazione reciproca[4], la correlazione o i punteggi di similarità/distanza per selezionare le caratteristiche di un *dataset*. L'obiettivo è quello di penalizzare una *feature* in base a quanto sia ridondante rispetto alle altre *feature* selezionate e premiarla in base a quanto sia pertinente. Vengono quindi selezionate *feature* che hanno un livello di pertinenza alto e un livello di ridondanza basso.

La massima rilevanza consiste nel ricercare le caratteristiche che soddisfano la formula definita in 6 definita dal valore medio di tutti i valori di informazione reciproca tra le singole caratteristiche x_i e la classe *target* (anch'essa una caratteristica) c.

$$maxD(S,c) = \frac{1}{|S|} \sum_{x_i \in S} I(f_i; c), \tag{6}$$

dove:

- $D(S,c)$ è la funzione di selezione delle caratteristiche,
- S è l'insieme di caratteristiche considerate,
- $|S|$ è il numero di elementi in S,
- x_i è la i-esima caratteristica considerata,
- $I(x_i;c)$ è l'informazione reciproca tra la caratteristica f_i e la classe *target* c.

[4]L'informazione reciproca (MI, dall'inglese *Mutual Information*) è una misura di correlazione non lineare tra due variabili. Essa quantifica la quantità di informazione che una variabile fornisce sull'altra e viceversa.

È probabile che le caratteristiche selezionate in base alla massima rilevanza possano avere una forte ridondanza, ovvero che la dipendenza tra queste caratteristiche possa essere molto elevata. Quando due caratteristiche dipendono molto l'una dall'altra, la corrispondente forza discriminatoria della classe non cambierebbe molto se una di queste venisse rimossa. Pertanto, in modo da selezionare le caratteristiche mutuamente esclusive, è bene aggiungere la seguente condizione di minima ridondanza (Min-Redundancy) 7:

$$minR(S), R = \frac{1}{|S^2|} \sum_{x_i, x_j \in S} I(x_i, x_j). \tag{7}$$

Il criterio che combina i due vincoli di cui sopra viene chiamato "minima-ridondanza-massima-rilevanza" (mRMR). Definiamo l'operatore $\Phi(D, R)$ per combinare D e R e consideriamo la seguente forma più semplice per ottimizzare D e R contemporaneamente, quello che otteniamo è il seguente indice:

$$max\Phi(D, R), \Phi = D - R. \tag{8}$$

Nella pratica, si possono utilizzare metodi di ricerca incrementale per trovare le caratteristiche quasi ottimali definite da Φ. Supponiamo di avere già l'insieme di caratteristiche S_{m-1}, con $m-1$ caratteristiche. Il compito è selezionare la caratteristica m-esima dall'insieme delle caratteristiche $\{X - S_{m-1}\}$. Ciò avviene selezionando la caratteristica che massimizza Φ. L'algoritmo incrementale ottimizza la seguente condizione:

$$\max_{x_j \in X - S_{m-1}} \left[I(x_j; c) - \frac{1}{m-1} \sum_{x_i \in S_{m1}} I(x_j; x_i) \right]. \tag{9}$$

L'obiettivo è riuscire a selezionare un insieme più ristretto di caratteristiche. Si selezionano quindi le caratteristiche con i punteggi di indice più alti, mentre quelle con punteggio più basso vengono scartate. Negli esperimenti (vedi Capitolo 3) la dimensione obbiettivo è stata impostata a 100 quindi il processo sopra descritto viene ripetuto 100 volte e ogni volta viene selezionata la caratteristica più rilevante, così fino a che si hanno 100 caratteristiche.

Boruta

Boruta [49] è un algoritmo di *feature selection* che utilizza un *wrapper* costruito sull'algoritmo di classificazione *random forest*. Quest'ultimo è un metodo *ensemble* (vedi Paragrafo 1.1.1) in cui la classificazione viene effettuata tramite una votazione di più classificatori, ovvero i *decision tree*. Questi alberi sono sviluppati in modo indipendente su diversi metodi di raccolta dei campioni del *training set*. Per capire l'importanza di una *feature* si tiene traccia della perdita di accuratezza della classificazione causata dalla permutazione casuale dei valori delle *feature*. Essa viene calcolata in maniera separata per tutti gli alberi della foresta che utilizzano una specifica *feature* per la classificazione. Successivamente vengono calcolate media e deviazione standard della perdita di accuratezza. In alternativa può essere usato *Z score* come misura di importanza, calcolato dividendo la perdita di accuratezza media per la deviazione standard. Purtroppo lo *Z score* non è direttamente correlato con la significatività statistica dell'importanza della *feature* restituita da *random forest* [89].

In Boruta, tuttavia, viene usato lo *Z score* poiché tiene conto delle fluttuazioni della perdita media di accuratezza tra gli alberi della foresta. Poiché però non è possibile usare direttamente lo *Z score* per misurare l'importanza, c'è la necessità di qualcosa di esterno per decidere l'importanza di una *feature*, cioè elementi che posso aiutare a distinguere l'importanza che può derivare da fluttuazioni casuali. A questo scopo vengono create delle *feature* che sono casuali per costruzione in quanto i valori sono ottenuti mescolando i valori della *feature* originale: chiameremo queste *feature* con il nome di *shadow feature*. Viene poi eseguita una classificazione utilizzando tutte le *feature*, comprese quelle *shawod*, e calcolata la loro importanza. L'importanza delle *shadow feature* viene usata come riferimento per decidere quali attributi sono importanti e quali no. Siccome la misura di importanza varia a causa dalla casualità del classificatore *random forest* è necessario ripetere la procedura di creazione delle *shadow feature* rimescolando i valori per ottenerne di nuove. Riassumendo Boruta si basa sulla stessa idea che costituisce il fondamento del classificatore *random forest*, ovvero: aggiungendo casualità al sistema e raccogliendo i risultati dall'insieme di campioni randomizzati si può ridurre l'impatto fuorviante dovuto alle fluttuazioni e dalle correlazioni casuali. In

questo caso, questa ulteriore casualità ci fornirà una visione più chiara di quali *feature* siano veramente importanti.

Di seguito sono stati illustrati i passaggi dell'algoritmo.

- Per ogni *feature* viene creata una caratteristica *"shadow"*, i cui valori si ottengono mescolando i valori della *feature* originale,
- le *feature shadow* generate vengono mescolate per rimuovere le loro correlazioni con l'etichetta del problema di apprendimento supervisionato che si sta affrontando,
- viene eseguito l'algoritmo *random forest* sul *dataset* contenente i dati di partenza più le *feature shadow*, i punteggi calcolati vengono salvati in Z,
- viene calcolato lo score Z massimo tra gli attributi *shadow* (MZSA),
- le *feature* con puntegggio significativamente inferiore a MZSA vengono considerate come "non importanti" e rimosse permanentemente dal *dataset*.
- le *feature* con importanza significativamente superiore a MZSA vengono considerate come "importanti",
- vengono rimossi tutte le *shadow feature*.
- il procedimento viene ripetuto fino a quando l'importanza viene assegnata a tutte le *feature*.

La complessità temporale della procedura, in casi realistici, è circa $O(P \cdot N)$, dove P e N sono rispettivamente il numero di *feature* e di oggetti. Questo può risultare dispendioso per grandi insiemi di dati; tuttavia, questo sforzo è essenziale per produrre una selezione statisticamente significativa di *feature* rilevanti.

The maximal information coefficient (MIC)

Il *Maximal Information Coefficient* (MIC) è una misura statistica usata per identificare nuove associazioni tra coppie di variabili, o *feature*. I valori del MIC sono compresi nell'intervallo 0 e 1. Se il valore è più vicino a 1, le due variabili hanno una relazione più stretta; se, invece, è più vicino a 0, le due variabili hanno

più probabilità di essere indipendenti. Il MIC appartiene a una classe più ampia di statistiche basate sulla *Maximal Information-based Nonparametric Exploration* (MINE) [90].

Intuitivamente, MIC si basa sul concetto che se esiste una relazione tra due variabili, allora è possibile tracciare una griglia sullo *scatter plot*[5] in modo da suddividere i dati per evidenziare la relazione. Pertanto, per calcolare il MIC di un insieme di dati a due variabili, si esplorano tutte le griglie calcolando per ogni coppia di interi (x, y) la maggiore *mutual information*[6] possibile raggiunta da qualsiasi griglia $x - by - y$ applicata ai dati [91]. Successivamente questi valori di *mutual information* vengono normalizzati per garantire un confronto equo tra griglie di dimensioni diverse. Definiamo la matrice caratteristica $M = (m_{x,y})$, dove $m_{x,y}$ è la *mutual information* normalizzata più alta raggiunta da qualsiasi griglia $x - by - y$, e la statistica MIC come il valore massimo in M. Il MIC è considerato una misura più robusta rispetto ad altre misure di correlazione, poiché tiene conto sia della quantità che della qualità della *mutual information* tra le variabili.

2.3.4 Intrinsic Dimensionality

Il concetto di *Intrinsic Dimensionanality*, o dimensionalità intrinseca in italiano, definito da Bennet [92] come il numero minimo di *feature* necessario per generare una descrizione dei dati mantenendo la struttura "intrinseca" che caratterizza l'insieme dei dati in modo da minimizzare la perdita di informazione. La stima della dimensione intrinseca può essere un compito difficile poiché quando il numero di *feature* è elevato i dati si comportano in modo controintuitivo a causa della *curse of dimensionality* (vedi Paragrafo 2.3.3). Un ulteriore problema risiede nel fatto che negli insiemi di dati reali non tutte le *feature* hanno la stessa importanza: alcune sono molte informative, altre possono essere classificate come "rumore", ovvero poco informative. In generale, la dimensione intrinseca viene calcolata utilizzando

[5]Il grafico di dispersione, detto anche *scatter plot*, è un tipo di grafico in cui due variabili di un *dataset* sono riportate su uno spazio cartesiano. I dati sono visualizzati tramite una collezione di punti ciascuno con una posizione sull'asse orizzontale determinato da una variabile e sull'asse verticale determinato dall'altra.

[6]La *mutual information*, o mutua informazione, di due variabili casuali è una misura di dipendenza tra due variabili casuali X e Y che quantifica la quantità di informazione comune che le due variabili condividono.

una serie di algoritmi che analizzano le proprietà geometriche e statistiche dei dati, come la distanza tra i punti o la loro distribuzione. Il calcolo della dimensione intrinseca può essere utilizzato per una vasta gamma di applicazioni, tra cui la riduzione delle dimensionalità, il *clustering* e la classificazione.

Intrinsic Dimension - Two Nearest Neighbor

L'*Intrinsic Dimension–Two Nearest Neighbor* (TWO-NN) è una tecnica per calcolare la dimensione intrinseca che utilizza solo le distanze ai primi due vicini più vicini di ogni punto nel *dataset* [93]. L'algoritmo è composto dai seguenti passi:

- per ogni coppia di punti si calcola la distanza,
- dato un generico punto si indica con r_1 e r_2 la più piccola e la seconda più piccola distanza precedentemente calcolata,
- per ogni punto i si calcola $\mu_i = \frac{r_2}{r_1}$,
- si calcola la cumulativa empirica $F^{emp}(\mu)$ ordinando i valori di μ attraverso una permutazione ρ che, dato un indice i, restituisce la posizione del punto i-esimo nella sequenza ordinata rispetto ai valori di μ. Si definisce quindi $F^{emp}(\mu_{\rho(i)}) \doteq \frac{i}{N}$,
- si calcola la regressione lineare a partire dai punti nel piano $\{(\log(\mu_i), -\log(1-F^{emp}(\mu_i))), |i = 1, ..., N|\}$ considerando solo rette passanti per l'origine. La pendenza della retta di regressione rappresenta la stima della dimensione intrinseca del *dataset*.

TWO-NN è una tecnica di stima efficiente e facile da implementare tuttavia, può essere influenzato dalle proprietà statistiche del *dataset*, come la presenza di *outlier* o di *cluster* di densità diversa.

Capitolo 3

Esperimenti

Nel Capitolo 2 è stato preso in esame l'importanza dei dati e di come essi devono essere modificati, modellati e puliti in modo che risultino più qualitativi per un determinato modello. In questo Capitolo viene approfondita la fase di *preprocessing* applicata ai dati. Successivamente vengono esaminate le diverse combinazioni di tecniche di riduzione della dimensionalità applicate ai dati prima di procedere con l'addestramento al fine di ottenere prestazioni più elevate possibili. Infine sono discussi i risultati ottenuti.

3.1 Preprocessing

Inizialmente le operazioni di *preprocessing* hanno richiesto di controllare se ci fossero valori nulli. Nello specifico sono state scartate *feature* con un numero di valori nulli superiore al 20%. Dopo questa rimozione non sono rimasti ulteriori valori di questo tipo nei dati quindi non è stato necessario adottare tecniche di imputazione (illustrate nel Paragrafo 2.2). Successivamente i dati sono stati normalizzati usando lo *scaler MinMaxScaler* del pacchetto di *scikit-learn*[1]. Questa tecnica trasforma le caratteristiche scalandole in un determinato intervallo [94], in questo caso tra 0 e 1. Per ottenere un dato normalizzato con la tecnica descritta precedentemente

[1]Scikit-learn è una libreria open source per l'apprendimento automatico che supporta l'apprendimento supervisionato e non supervisionato. Fornisce inoltre vari strumenti per l'adattamento dei modelli, la preelaborazione dei dati, la selezione dei modelli, la loro valutazione e molte altre utilità. Link del pacchetto: https://scikit-learn.org/stable/.

viene applicata la seguente formula:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{norm}}. \tag{10}$$

In seguito i *dataset* (proteine, miRNA e mRNA) sono stati filtrati dalle caratteristiche, o variabili, che avevano una variabilità estremamente bassa, ovvero 0.05. Questo processo viene effettuato poiché caratteristiche con così bassa variabilità possono essere considerate non informative quindi si preferisce scartarle sia per snellire il *dataset*, con conseguente abbassamento del costo computazionale, sia perché questo comporta un incremento della qualità dei dati. In Tabella 2 è possibile vedere le dimensioni prima e dopo il filtraggio delle variaibli poco informative.

	prima	dopo	*feature* scartate
proteins	216	216	0
mRNA	20501	18465	2036
miRNA	1046	773	273

Tabella 2: Tabella che mostra le dimensioni prima e dopo aver filtrato le caratteristiche con variabilità inferiore a 0.05.

3.2 Model selection

Il processo di *model selection* in *machine learning* è la selezione della configurazione di iperparametri dell'algoritmo di apprendimento che più si adatta ai dati a disposizione. È composto da tre fasi: selezione del modello, nel mio caso *random forest*, regolazione degli iperparametri del modello, discusso nel Paragrafo 3.2.1 e valutazione delle prestazioni, ovvero dei risultati, discussi nel Paragrafo 3.5.

3.2.1 Tuning degli iperparametri

Il *tuning* degli iperparametri negli esperimenti è stato effettuato usando una *GridSearchCV* fornito dal pacchetto *scikit-learn* [95]. *GridSearchCV* effettua una ricerca esaustiva dei valori degli iperparametri del modello prendendo i valori da una griglia fornita in *input*. La metrica usata per decidere quale configurazione di iperparametri fosse la migliore tra tutte quelle possibili rispetto al *test set* è stata l'AUPRC, discussa nel Paragrafo 3.4. In pratica *GridSearchCV* cerca la configurazione di iperparametri che massimizzano il valore di AUPRC calcolato sul *test set*. Negli esperimenti è stata usata la griglia in Figura 17.

```
model_selection_grid_DT = [
    {'criterion': ['gini', 'entropy'],
     'max_leaf_nodes': [None, 2, 5, 10],
     'max_features': [None, 'sqrt', 'log2'],
    'n_estimators': [51, 101, 251, 500]}
]
```

Figura 17: Griglia di iperparametri usata all'interno della fase di *tuning* degli iperparametri mediante la *GridSearchCV*.

La valutazione della configurazione degli iperparametri è stata fatta usando una *cross-validation* interna a *GridSearchCV*. Nello specifico *GridSearchCV* opera una *Stratified k fold cross-validation* (vedi Paragrafo 3.3) usando un numero di *fold* decisi dall'utente. In particolare sono stati usati 2 *fold* poiché il costo computazionale, alzando il numero di *fold* interni, si alzava enormemente avendo già 10 *fold* esterni (vedi Paragrafo successivo).

3.3 Cross-validation

Come discusso nel Paragrafo 1.1.1 esistono diverse strategie per effettuare *cross-validation*. Negli esperimenti è stata usata una versione diversa da quelle spiegate; in particolare è stata utilizzata una *Stratified K-Folds cross-validation* sia come *cross-validation* "esterna", per stimare quindi la *performance* di un modello e quindi la bontà di generalizzazione, sia "interna" (vedi Paragrafo 3.2.1). La scelta di questo tipo specifico di *cross-validation* è ritenuta opportuna in quanto si adatta

molto bene quando si hanno a disposizione dati sbilanciati, esattamente il nostro caso, come visibile in Figura 13. Tale tecnica consiste nel dividere il *dataset* in k sotto-insiemi (chiamati *fold*) mantenendo però le proporzioni delle diverse classi presenti nell'insieme di dati. Questo garantisce che tutte le categorie del problema di classificazione vengano rappresentate in ogni *fold* di addestramento e di *test*. A ogni iterazione, un diverso *fold* viene utilizzato come *test set* e gli altri $k-1$ come *training set* per addestrare il modello. Questo processo viene ripetuto k volte, utilizzando un diverso *fold* come *test set* ogni volta. La valutazione del modello viene poi calcolata come media delle performance su tutti i k-*fold*. Nello specifico è stata usata una *Stratified 10-fold cross validation* (vedi Figura 1).

3.4 Metrica di performance

Come metrica di performance è stata usata l'area sotto la curva *Precision-Recall* (AUPRC). Questa metrica è comunemente utilizzata in problemi di classificazione binaria, sopratutto in casi in cui la distribuzione delle classi è sbilanciata [96], ovvero il nostro caso, come visibile in Figura 13. La curva *Precision-Recall* mostra la relazione che sussiste tra la *precision* e la *recall*, la quali sono spiegate nel Paragrafo 1.1.1, per diverse soglie di classificazione. Per calcolare la curva *Precision-Recall*, prima si eseguono le predizioni utilizzando un modello di classificazione. Poi, per ogni soglia di classificazione, si calcolano *precision* e *recall*. Si ottiene così una serie di coppie di *precision-recall* per diverse soglie di classificazione. Queste coppie possono essere visualizzate come punti su un piano cartesiano, dove la *recall* viene mostrata sull'asse delle x e la *precision* viene mostrata sull'asse delle y. Tracciando una linea che connette tutti i punti, si ottiene la curva *Precision-Recall*. In Figura 21 è possibile vederne un esempio.

Un'area elevata sotto la curva rappresenta sia una elevata *recall* sia un'elevata *precision*, dove un'elevata *precision* si riferisce a un basso tasso di falsi positivi e un'elevata *recall* a un basso tasso di falsi negativi. Punteggi elevati per entrambe le metriche indicano che il classificatore restituisce risultati accurati (alta precisione), oltre a restituire la maggior parte di tutti i risultati positivi (alto richiamo).

In genere, è importante avere un equilibrio tra *precision* e *recall* in modo da ottenere una buona *performance* del modello. Tuttavia, a volte è più importante

dare maggiore importanza a una delle due metriche in base alle esigenze specifiche del problema di classificazione. Ad esempio, in un problema medico potrebbe essere più importante identificare il più possibile i veri positivi (alte *recall*), anche a discapito di una *precision* più bassa. In altro contesto, come ad esempio la classificazione di email come spam, potrebbe essere più importante la *precision*, in modo da evitare di classificare come spam email importanti.

Poiché la curva *Precision-Recall* non utilizza i veri-negativi, essi non influenzeranno i risultati. È possibile quindi utilizzare dati sbilanciati come 98% di esempi negativi e 2% di esempi positivi e il modello si concentrerà su come valuta il 2% di esempi positivi. Se il modello riesce a gestire bene gli esempi positivi allora il valore di AUPRC sarà alto, altrimenti sarà basso. L'AUPRC varia su una scala da 1 a 0. In contesti reali, in particolare quelli medici, la frazione delle osservazioni positive è spesso inferiore a 0.5, il che significa che il valore di AUPRC sarà di base inferiore rispetto a quello di AUROC (vedi Paragrafo 1.1.1). È possibile, per esempio, ottenere un AUROC pari a 0.8 e un AUPRC pari a 0.3. Ovviamente a prima vista sembrerebbe meglio ottenere prestazioni di 0.8, anche se il numero più significativo per il problema in questione potrebbe benissimo essere l'AUPRC di 0.3.

È un po' più difficile interpretare l'AUPRC rispetto all'AUROC, questo perché la *baseline* per AUROC sarà sempre 0.5, un classificatore casuale, o un lancio di moneta, porterà a un AUROC di 0.5. Con AUPRC, invece, la *baseline* è pari alla frazione di positivi [97], calcolata come numeri di esempi positivi / numero totali di esempi. Ciò significa che classi diverse hanno *baseline* di AUPRC diverse. Una classe con il 12% di positivi ha un AUPRC di base di 0.12, quindi ottenere un AUPRC di 0.40 su questa classe è ottimo. Tuttavia, una classe con il 98% di positivi ha un AUPRC di base di 0.98, il che significa che ottenere un AUPRC di 0.40 su questa classe è uno scarso risultato. In questo caso specifico la *baseline* per i dati è 0.122, ottenuta prendendo tutte le osservazioni positive, ovvero 77, diviso per il numero di osservazioni totali, ovvero 627.

Il calcolo del valore di AUPRC per la valutazione della bontà di generalizzazione del modello è stato effettuato in maniera duplice:

- "interna": all'interno della *Stratified 10-fold cross validation* a ogni iterazione

viene calcolato un valore di AUPRC, salvato in una lista e, al termine delle 10 iterazioni, fatta la media; chiameremo questo tipo di AUPRC "interna" poiché calcolata, per 10 volte, internamente ai *fold*;

- "esterna": a ogni iterazione della *Stratified 10-fold cross validation* le predizioni effettuate dal modello sono salvate in una lista e concatenate con le predizioni delle precedenti iterazioni fino a raggiungere le 10 iterazioni date dai 10 fold. Successivamente le predizioni concatenate sono confrontate con le etichette reali, anch'esse concatenate a ogni iterazione, in maniera complessiva, calcolando così il valore di AUPRC "esterno", ovvero calcolato esternamente ai *fold*.

Gli esperimenti effettuati sono stati una combinazione di tecniche di *dimensionality reduction* come *feature selection*, *feature extraction* o altre tecniche come *t-SNE* (vedi Paragrafo 2.3.1), applicati singolarmente o una a seguito dell'altra sui vari *dataset* illustrati nel Capitolo 2, quali: proteine, CNV, miRNA e mRNA presi singolarmente, concatenazione di tutti e quattro (*Whole_dataset*) o concatenazione di alcuni (proteine+miRNA e proteine+miRNA+mRNA). Di seguito sono stati illustrati i risultati ottenuti con particolare attenzione a quelli migliori. Il modello utilizzato è stato sempre *random forest*, illustrato nel Paragrafo 1.1.1.

3.5 Risultati ottenuti

Nelle tabelle che seguono, ogni riga corrisponde a una tecnica di *dimensionality reduction* o alla concatenazione di più tecniche applicate una a seguito dell'altra. Il primo valore indica l'AUPRC calcolata "internamente" mentre il secondo "esternamente" (vedi Paragrafo precedente). Le colonne indicano invece i vari *dataset* utilizzati negli esperimenti illustrati nel Capitolo 2 con l'aggiunta di *Whole_dataset*, concatenazione di tutti i dataset precedenti, dopo aver normalizzato i dati e scartato le *feature* con una variabilità estremamente bassa, ovvero 0.05. La fase di filtraggio delle *feature* descritta in precedenza non è stata però applicata al dataset CNV.

La Tabella 3 che segue mostra gli esperimenti effettuati sui diversi *dataset*, mentre in Figura 18 è possibile vedere il grafico delle curve di *precision/recall*

calcolate sui 10 *fold*, quindi "internamente", del miglior risultato ottenuto, usando il *dataset* mRNA e come tecniche *Mannwhitney+Umap* usando 50 come numero di *feature* estratte attraverso la seconda tecnica. Usando la tecnica di *feature selection* *mRMR* sono invece state selezionate 100 *feature*.

	miRNA	Proteins	mRNA	Whole_dataset	CNV
mannwhitneyu	**0.25**, **0.16**	**0.23**, **0.15**	**0.23**, 0.15	**0.23**, **0.16**	**0.13**,0.10
mannwhitneyu_with_umap	**0.27**, 0.12	**0.26**, 0.12	**0.40**, **0.14**	0.**21**, 0.11	**0.27**, **0.15**
boruta	**0.18**, **0.14**	**0.17**, 0.12	**0.20**, **0.16**	**0.22**, **0.16**	-1
mic	**0.25**, **0.18**	**0.21**, **0.16**	**0.23**, **0.17**	-1	**0.15**, 0.11
mrmr	**0.20**, **0.14**	**0.19**, **0.14**	-1	-1	-1
spearman	**0.17**, **0.16**	**0.21**, **0.14**	**0.18**, **15**	**0.19**, **0.20**	**0.13**, 0.10

Tabella 3: Tabella che riassume i valori di AUPRC calcolati sia "internamente", primo valore, che "esternamente", secondo valore. Dove è presente -1 significa che, utilizzando le corrispondenti tecniche, il calcolo non è mai arrivato a convergenza dopo giorni di calcolo, si è deciso quindi di fermare la computazione. In arancione è evidenziato il valore massimo ottenuto mentre in grassetto tutti i valori maggiori della *baseline* di 0.12.

In questo esperimento la migliore configurazione di iperparametri selezionata dalla *GridSearch* è la seguente:

- *max_leaf_nodes* = 10,
- *n_estimators* = 51.

Successivamente si sono svolti ulteriori esperimenti: questa volta inizialmente applicando una tecnica di *feature selection* utilizzando come metrica *Pearson* (vedi Paragrafo 2.3.3) con un *cut-off* di 0.8. In Tabella 4 è possibile vedere le dimensioni prima e dopo l'applicazione di *Pearson*.

Nella Tabella 5 è, invece, possibile vedere i risultati ottenuti.

In questi esperimenti la tecnica di *feature selection*, usando come metrica *Pearson*, è stata applicata al *dataset* completo prima di essere dato in *input* alla funzione che applica una *Stratified 10-fold cross validation*. Successivamente le restanti tecniche di *feature selection* e *dimensionality reduction* sono state applicate *on-fold*, ovvero internamente al *fold* generato. Nello specifico per quanto riguarda il calcolo della dimensionalità intrinseca viene effettuato attraverso l'algoritmo TWO-NN

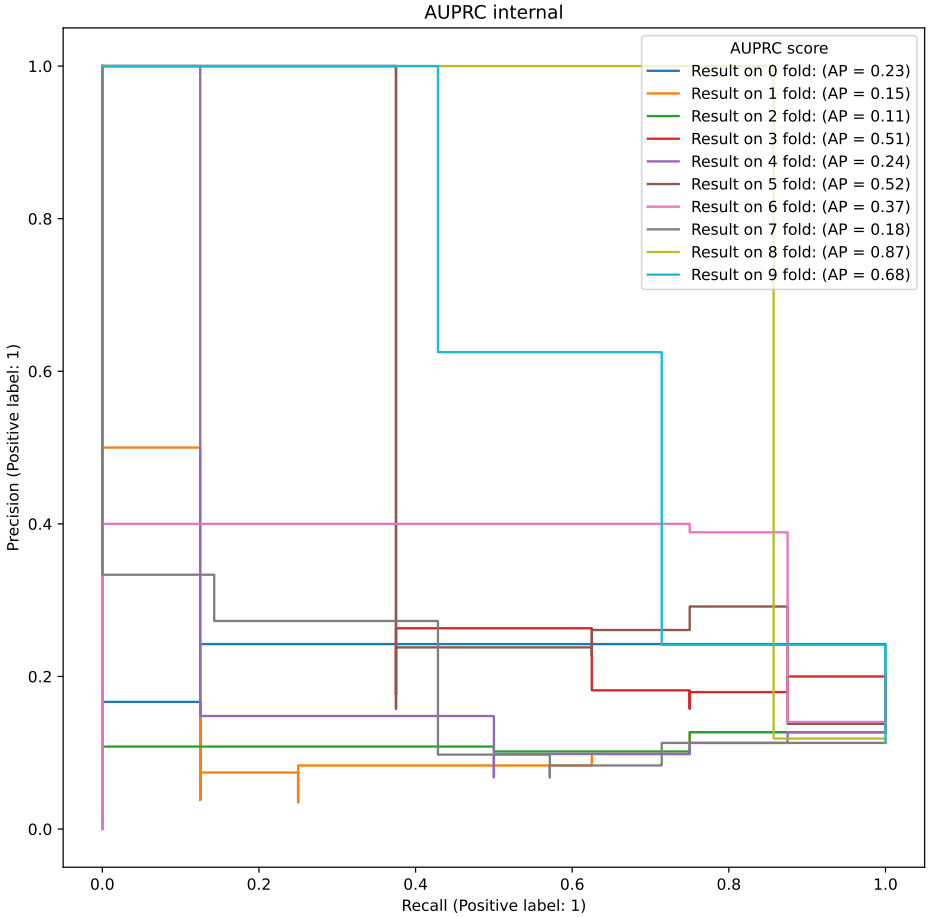


Figura 18: Grafico delle curve di *precision/recall* calcolate sui 10 *fold*, quindi "internamente", del miglior risultato ottenuto, usando il *dataset* mRNA e come tecniche *Mannwhitney+Umap*.

(vedi Paragrafo 2.3.4) applicato sul *training set* del *fold*. La dimensione ottenuta viene poi utilizzata come dimensione obiettivo per il calcolo di *UMAP* e *t-SNE* sia del *training set* che del *test set*.

Per quanto riguarda invece gli altri 3 esperimenti:

- viene applicato *Mannwhitney* sempre *on fold* e le dimensioni selezionate a

	prima	dopo	*feature* scartate
proteins	216	216	0
mRNA	18465	18411	54
miRNA	773	761	12
CNV	24776	194	24582
Whole_dataset	44230	20386	23844
proteins+miRNA	989	977	12
proteins+miRNA+mRNA	19454	19344	110

Tabella 4: Tabella che mostra le dimensioni prima e dopo aver filtrato le *feature* usando *Pearson* come indice di correlazione per fare *feature selection*.

	miRNA	Proteins	mRNA	Whole_dataset	CNV	Prot+miRNA	Prot+miRNA+mRNA
P+intr+umap	**0.38**, **0.18**	**0.35**, **0.14**	**0.32**, **0.21**	**0.27**, **0.17**	**0.20**, **0.15**	**0.43**, **0.14**	**0.38**, **0.13**
P+intr+tnse	**0.16**, 0.12	**0.20**, **0.14**	**0.19**, **0.13**	**0.16**, **0.14**	**0.16**, **0.11**	**0.18**, **0.12**	**0.16**, 0.12
P+mann+boruta	**0.17**, **0.13**	**0.22**, **0.15**	**0.23**, **0.18**	**0.22**, **0.15**	-1	**0.16**, **0.13**	**0.21**, **0.15**
P+mann+mrmr	**0.23**, **0.17**	**0.21**, **0.16**	-1	-1	**0.16**, 0.11	**0.22**, **0.16**	-1
P+mann	**0.23**, **0.17**	**0.21**, **0.15**	**0.21**, **0.16**	**0.22**, **0.17**	**0.16**, 0.12	**0.25**, **0.17**	**0.24**, **0.18**

Tabella 5: Tabella che riassume i valori di AUPRC calcolati sia "internamente", primo valore, che "esternamente", secondo valore. Dove è presente -1 significa che, utilizzando le corrispondenti tecniche, il calcolo non è mai arrivato a convergenza dopo giorni di calcolo, si è preferito quindi fermare la computazione. *P* sta per *Pearson* (vedi Paragrafo 2.3.3), *intr* indica l'applicazione della *intrinsic dimensionality* (vedi Paragrafo 2.3.4), *mann* indica l'applicazione di *Mannwhitney* (vedi Paragrafo 2.3.3), la prima tecnica è stata applicata all'intero *dataset* mentre le restanti sono state applicate *on fold*. In arancione è evidenziato il valore massimo ottenuto mentre in grassetto tutti i valori maggiori della *baseline* di 0.12.

ogni iterazione vengono salvate per essere usate nei restanti due esperimenti;

- alle *feature* selezionate da *Mannwhitney* viene applicato in cascata *boruta* (vedi Paragrafo 2.3.3) sempre *on fold*;
- alle *feature* selezionate da *Mannwhitney* viene applicata in cascata *mRMR* (vedi Paragrafo 2.3.3) sempre *on fold*, usando 100 *feature* come obiettivo.

In Figura 19 è possibile vedere le diverse curve di *precision-recall* calcolate

sui 10-*fold* per il miglior risultato ottenuto ovvero usando il *dataset* concatenato così composto: proteine+miRNA. Le tecniche di riduzione della dimensionalità sono invece state: Pearson+calcolo dimensionalità intrinseca+umap a seguire. In questo esperimento la miglior configurazione di iperparametri tra i 10 diversi addestramenti effettuati nei 10 fold è:

- *criterion* = *entropy*,
- *max_features* = log_2,
- *max_leaf_nodes* = 5,
- *n_estimators*=51.

Infine sono stati eseguiti ulteriori esperimenti: la tipologia è identica a quella illustrata sopra ma questa volta tutte le tecniche di *feature selection* e *dimensionality reduction* sono state applicate sul *dataset* completo, non sul *fold*. In Tabella 4 è possibile vedere le dimensioni prima e dopo l'applicazione di *Pearson*.

In Tabella 6 è possibile vedere i risultati. Nelle Figure 21 e 22 invece è possibile vedere la curva *precision/recall* del migliore addestramento ottenuto rispettivamente "esternamente" e "internamente".

	miRNA	Proteins	mRNA	Whole_dataset	CNV	Prot+miRNA	Prot+miRNA+mRNA
P+intr+umap	**0.78**, **0.78**	**0.71**, **0.70**	**0.63**, **0.64**	**0.68**, **0.66**	**0.55**, **0.56**	**0.74**, **0.74**	**0.70**, **0.68**
P+intr+tnse	**0.21**, **0.15**	**0.14**, 0.11	**0.17**, 0.11	**0.16**, **0.14**	**0.15**, 0.12	**0.19**, **0.14**	**0.18**, **0.14**
P+mann+boruta	**0.29**, **0.18**	**0.24**, **0.17**	**0.38**, **0.29**	**0.39**, **0.30**	-1	**0.30**, **0.24**	**0.38**, **0.32**
P+mann+mrmr	**0.25**, **0.18**	**0.23**, **0.17**	-1	-1	**0.29**, **0.16**	**0.29**, **0.22**	-1
P+mann	**0.22**, **0.18**	**0.25**, **0.19**	**0.26**, **0.18**	**0.26**, **0.16**	**0.24**, **0.17**	**0.31**, **0.21**	**0.25**, **0.17**

Tabella 6: Tabella che riassume i valori di AUPRC calcolati sia "internamente", primo valore, che "esternamente", secondo valore. Dove è presente -1 significa che, utilizzando le corrispondenti tecniche, il calcolo non è mai arrivato a convergenza dopo giorni di calcolo, e si è preferito quindi fermare il calcolo. *P* sta per *Pearson* (vedi Paragrafo 2.3.3), *intr* indica l'applicazione della *intrinsic dimensionality* (vedi Paragrafo 2.3.4), *mann* indica l'applicazione di *Mannwhitney* (vedi Paragrafo 2.3.3), tutte le tecniche sono state applicate sull'intero *dataset*. In arancione è evidenziato il valore massimo ottenuto mentre in grassetto tutti i valori maggiori della *baseline* di 0.12 (vedi Paragrafo 3.4).

In Figura 20 è possibile vedere la matrice di confusione del miglior risultato ottenuto "esternamente" (0.78). Essa mostra come il classificatore non sbagli mai

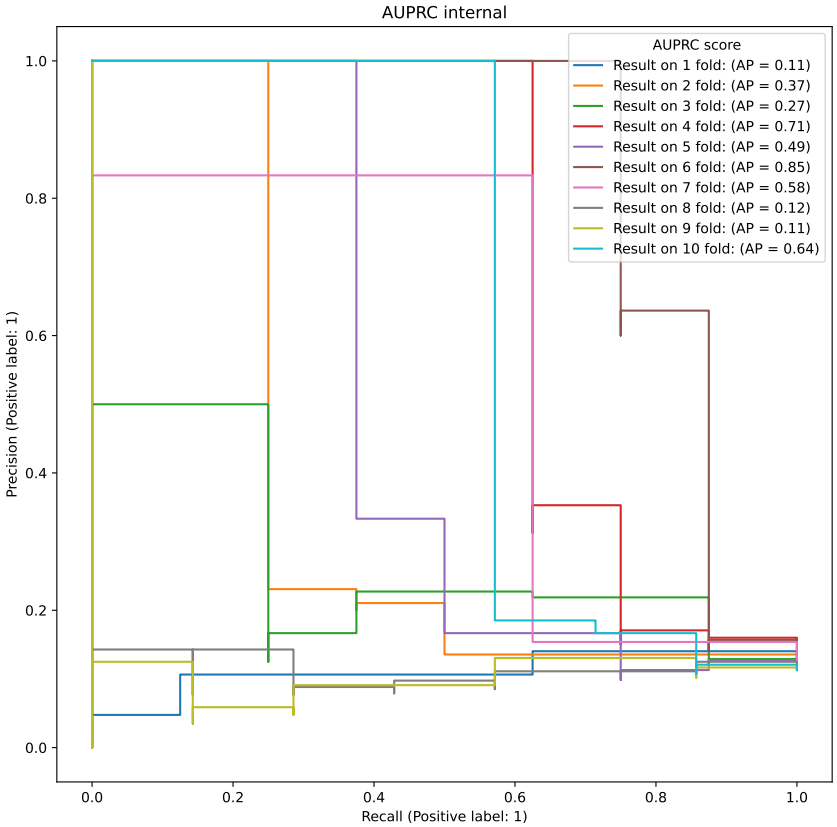


Figura 19: Grafico delle curve di *precision/recall* calcolate sui 10 *fold*, quindi "internamente", del miglior risultato ottenuto, usando il *dataset* proteine+miRNA e come tecniche *Pearson*+calcolo dimensionalità intrinseca+umap.

a classificare pazienti sani (su 550 pazienti sani vengono classificati tutti come tali). Per quanto riguarda invece i pazienti malati, su 77 totali, 23 (30%) vengono classificati sani (basso sinistra), sbagliando, mentre 54 (70%) vengono classificati malati (basso sinistra), non sbagliando. In Figura 23 è possibile vedere invece il calcolo della AUROC (vedi Paragrafo 1.1.1).

Infine in Tabella 7 è possibile vedere valori delle metriche *precision*, *recall*, *F-1-score* per le classi 0 e 1, ovvero rispettivamente sano e malato. È inoltre presente la metrica *support* che rappresenta il numero di esempi di una classe specifica nel

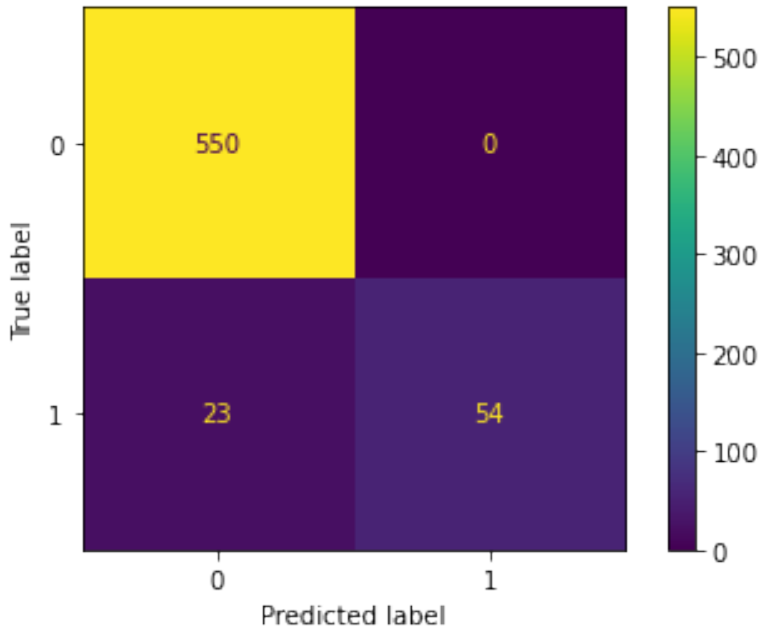


Figura 20: Matrice di confusione del miglior risultato di AUPRC calcolato "esternamente" ottenuto dagli esperimenti (0.78). Sull'asse x troviamo le predizioni fatte dal classificatore mentre sull'asse y troviamo le etichette corrette. 1 indica un paziente malato, 0 un paziente sano.

dataset. In questo caso quindi ci sono 550 esempi negativi e 77 positivi.

Inoltre questo addestramento ha un valore di *specificity* pari a 1.0. Per quanto riguarda invece la configurazione di iperparametri la migliore è la seguente:

- *max_features* = None,
- *max_leaf_nodes* = 10,
- *n_estimators*=251.

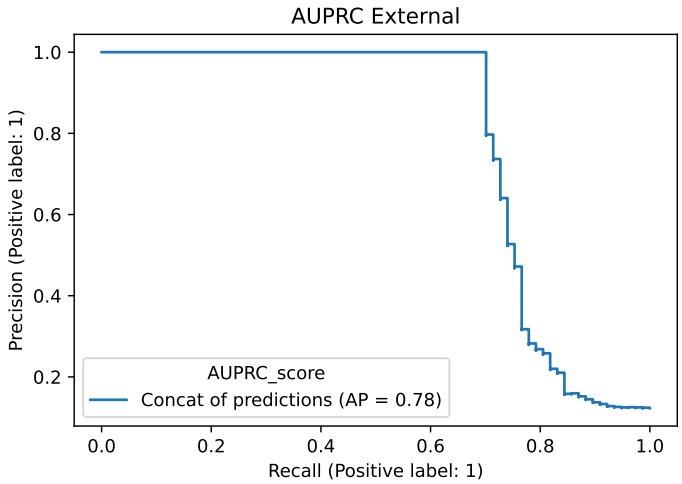


Figura 21: Grafico della curva di *precision/recall* calcolata come media delle AUPRC sui 10 *fold* quindi "esternamente", del miglior risultato ottenuto, usando il *dataset* miRNA e come tecniche *Pearson*, calcolo della dimensionalità intrinseca e la dimensione ottenuta (45) usata nella funzione che calcola *UMAP* a tutto il *dataset*.

	Precision	Recall	f1-score	support
class 0	0.96	1.00	0.98	550
class 1	1.00	0.70	0.82	77

Tabella 7: Tabella che mostra i valori di *precision, recall, f1-score e support*. *Class 0* indica la classe dei pazienti sani mentre *class 1* indica la classe dei pazienti malati.

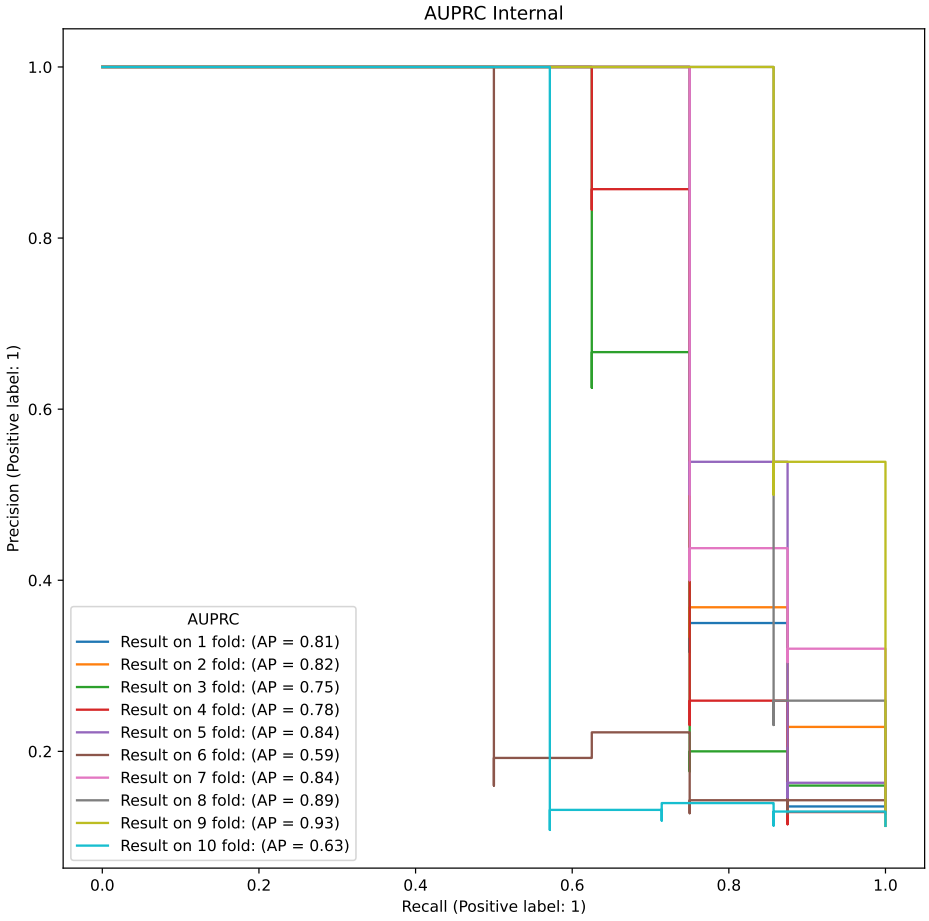


Figura 22: Grafico delle curve di *precision/recall* calcolate sui 10 *fold*, quindi "internamente", del miglior risultato ottenuto, usando il *dataset* miRNA e come tecniche *Pearson*, calcolo della dimensionalità intrinseca e la dimensione ottenuta usato nella funzione che calcola *UMAP* a tutto il *dataset*.

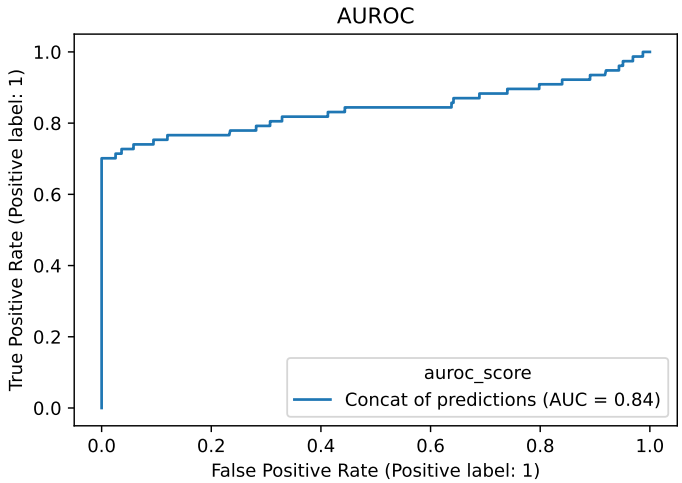


Figura 23: Grafico della curva ROC, anch'essa calcolata "esternamente", del miglior risultato ottenuto, usando il *dataset* miRNA e come tecniche *Pearson*, calcolo della dimensionalità intrinseca e la dimensione ottenuta (ovvero 45) usata nella funzione che calcola *UMAP* a tutto il *dataset*.

3.6 Analisi dei risultati

Osservando la Tabella 3 è possibile vedere che i primi esperimenti non presentano un valore di AUPRC elevato. Va però ricordata l'osservazione sulla *baseline* fatta nel Paragrafo 3.4. Come detto precedentemente in questo caso la *baseline* è di 0.12 quindi valori superiori a questa quantità sono da considerarsi positivi. In ogni caso non c'è stato un esperimento con esito estremamente positivo. Il valore massimo di AUPRC ottenuto è stato 0.40 usando come *dataset* mRNA e come tecniche *Mannwhitney* e *UMAP* a seguire, il valore è stato ottenuto calcolando l'AUPRC internamente. In Tabella 3 sono visibili anche valori pari a -1, questo indica che il calcolo non è mai riuscito a convergere. Ciò è causato dall'alto numero di dimensioni del dataset mRNA, circa 20.000 e di conseguenza da *Whole_dataset*, composto dalla concatenazione di tutti i *dataset*.

Per quanto riguarda gli esperimenti effettuati i cui esiti sono visibili in Tabella 5 i valori di AUPRC sono mediamente più elevati rispetto agli esperimenti precedenti. Questi esperimenti sono stati effettuati applicando prima *Pearsom* sull'intero *dataset*. Successivamente è stata calcolata la dimensionalità intrinseca e la dimensione ottenuta usata come dimensione obiettivo per UMAP, queste tecniche sono state applicate *on fold*. L'addestramento migliore ha portato a un valore di AUPRC di 0.43 usando il *dataset* proteine+miRNA. Anche in questo caso il valore viene calcolato "internamente". In Tabella 5, come nella precedente, è possibile trovare valori pari a -1 e questo è dovuto allo stesso motivo sopra citato.

Infine, in Tabella 6, è possibile vedere i risultati ottenuti usando le tecniche sull'intero *dataset* e non *on fold*. Complessivamente sono i risultati migliori di tutti gli esperimenti e in particolare troviamo ottimi risultati ottenuti usando come tecniche: *Pearson+intrinsic dimensionality* e *UMAP* a seguire. Su tutti i *dataset* questa combinazione porta a ottimi risultati, quello migliore in assoluto è stato ottenuto usando come *dataset* miRNA ottenendo 0.78 e 0.78, calcolando l'AUPRC, rispettivamente, "internamente" ed "esternamente". In Figura 20 è mostrata la matrice di confusione dell'addestramento il cui valore di AUPRC è calcolato "esternamente". Essa mostra come il classificatore classifichi come sani tutti i pazienti che effettivamente lo sono (550 su 550). Per quanto riguarda invece i malati, su 77 pazienti: 54 (ovvero il 70%) vengono classificati come malati, in maniera corretta,

mentre i restati 23 pazienti vengono classificati come sani, sbagliando.

Complessivamente possiamo accorgerci di come il calcolo dell'AUPRC eseguito "internamente" porti a valori più elevati nella maggior parte dei casi. Inoltre è bene notare come in tutte le tabelle i risultati migliori siano sempre ottenuti usando *UMAP* come tecnica di riduzione della dimensionalità.

3.7 Tecnologie usate

Per eseguire questo studio sono state utilizzate diverse tecnologie, prima tra tutte il linguaggio usato è stato *Python* 3.10.6, unitamente a diversi pacchetti specifici per l'analisi dei dati quali: *pandas* e *numpy*. Pacchetti per applicare le tecniche di riduzione della dimensionalità quali: *scipy*, *umap*, *minepy*, *pymrmr* e *boruta*. Infine *sklearn* per l'applicazione specifica di tecniche di apprendimento automatico. Come ambiente di lavoro è stato usato *Jupyter Notebook*, mentre come tool di *versioning*, *git* unitamente a *github*.

Capitolo 4

Conclusioni

In questo lavoro è stato sviluppato un modello di predizione usando dati provenienti dai geni BRCA in modo da predire diagnosi e prognosi del carcinoma mammario invasivo per nuovi pazienti. In particolare si voleva analizzare se l'utilizzo di dati multi-omici aumentasse la precisione di classificazione usando l'algoritmo di apprendimento supervisionato *random forest*. La scelta è ricaduta su questo algoritmo poiché famoso in letteratura per essere uno dei migliori quando si utilizzano dati eterogenei perché utilizza una foresta di *decision tree* in cui ogni albero è formato da decisioni basate su diverse caratteristiche.

Inizialmente sono state studiate e visionate alcune soluzioni simili in campo medico. In particolare studi sull'applicazione di apprendimento supervisionato per la predizione di malattie associate a varianti non codificanti e riguardanti la previsione di rischio di COVID-19 nei pazienti [33, 36].

I dati usati sono stati presi da *The Cancer Genome Atlas* (TCGA), programma scientifico di riferimento per la genomica del cancro. Per quanto riguarda invece le etichette sono state prese da un *dataset* curato manualmente, noto come TCGA-CDR. Esso indica la presenza o assenza di evento tumorale in un paziente. I dati forniti contenevano valori nulli. Pertanto sono state eliminate le *feature* che contenevano un numero di valori nulli superiore al 20%, dopo questa scrematura non è stato necessario adottare tecniche di imputazione poiché non sono rimasti ulteriori dati di questo tipo. Successivamente i dati sono stati normalizzati a

un valore compreso tra 0 e 1. Infine sono state scartate *feature* con una variabilità inferiore a 0.05 poiché poco informative. A seguito di questa prima pulizia dei dati, sono state applicate tecniche di riduzione della dimensionalità poiché un numero di *feature* troppo elevato all'interno dei *dataset* porta a un significativo rallentamento degli algoritmi di apprendimento automatico, peggioramento della *performance* predittiva degli stessi algoritmi e difficoltà nell'interpretazione del modello. Queste tecniche sono state applicate sia singolarmente (vedi Tabella 3) che concatenando più tecniche (vedi Tabelle 5 e 6). Le tecniche sono state applicate sia ai singoli *dataset* (*proteins, miRNA, mRNA e CNV*), sia a *dataset* derivati dalla concatenazione dei precedenti (dati multi-omici come: *whole_dataset, proteins+miRNA e proteins+miRNA+mRNA*). La metrica usata sia per valutare la bontà di generalizzazione del modello sia per eseguire il *tuning* degli iperparametri è stata l'AUPRC (*Area Under the Precision-Recall Curve*). Questa metrica è comunemente utilizzata in problemi di classificazione binaria, sopratutto quando la distribuzione dei dati è sbilanciata, ovvero esattamente il nostro caso, come visibile in Figura 13. Il valore di AUPRC varia su una scala da 0 a 1. In contesti medici, come quello trattato in questo lavoro, la frazione delle osservazioni positive, ovvero che rappresentano pazienti malati, è spesso inferiore a 0.5, in questo caso è 0.12. Questo comporta che la *baseline* di AUPRC, per questo problema di classificazione, è esattamente 0.12 e valori superiori possono considerarsi buoni.

La maggior parte dei risultati ottenuti hanno un valore di AUPRC superiore alla *baseline* di 0.12 imposta da come è composto il *dataset* (550 pazienti sani, 77 pazienti malati). Non c'è però evidenza che l'utilizzo di dati multi-omici aumenti la capacità dell'algoritmo *random forest* di classificare in maniera più precisa pazienti sani e pazienti malati. A fronte di questo però ci sono tecniche che ottengono un buon valore di AUPRC, ovvero *Pearson* + calcolo dimensionalità intrinseca usando *Two Nearest Neighbor* (Two-NN) + UMAP impostando come numero di *feature* estratte quello calcolato da Two-NN. Questa concatenazione di tecniche ottiene i risultati migliori su ogni tipo di *dataset*, come si può vedere nelle Tabelle 5 e 6. La differenza tra gli esperimenti delle due tabelle risiede in come sono state applicate le tecniche di riduzione della dimensionalità. Nella prima le tecniche sono state applicate *on fold* mentre nella seconda sono state applicate sull'intero *dataset*. In particolare è applicando le tecniche sull'intero *dataset* e poi usando il

dataset ottenuto come *input* per l'algoritmo di apprendimento che si ottengono i risultati migliori, visibili in Tabella 6. Considerando la *baseline* di partenza, ovvero 0.12, un risultato di 0.78, utilizzando miRNA, è da considerarsi ottimo. In Figura 20 è possibile però vedere come il modello classifichi correttamente i dati di *test* dei pazienti sani, ma che su 77 malati, 54 vengono classificati come effettivamente malati mentre 23 pazienti vengono classificati come sani, sbagliando. In campo medico questo fatto non è da considerarsi propriamente ottimo poiché quasi il 30% dei malati vengono classificati non malati, quando in verità lo sarebbero.

4.1 Sviluppi futuri

Questo lavoro presenta alcune limitazioni intrinseche. Prima di tutto scarsità di osservazioni positive dovute allo sbilanciamento tra pazienti sani e malati. Alcune delle possibili soluzioni potrebbero essere quelle di raccogliere ulteriori osservazioni riguardanti pazienti positivi al carcinoma mammario invasivo anche se in campo medico non è una pratica così facile da realizzare. Un'altra soluzione potrebbe essere quella di adottare tecniche di bilanciamento delle classi, come *Synthetic Minority Over-Sampling Technique* (SMOTE) [98]. Inoltre usando alcune tecniche di riduzione della dimensionalità la computazione non è mai arrivata al termine dopo giorni di calcolo, ovvero in corrispondenza dei -1 visibili nelle Tabelle 5 e 6. L'*hardware* su cui sono stati condotti gli esperimenti è stato un processore molto recente, ovvero un Intel Core i5-13600K [99]. Tuttavia, per comprendere pienamente le potenzialità delle tecniche analizzate, sarebbe opportuno, come sviluppo futuro, quello di provare a eseguire ulteriori esperimenti su *hardware* ancora più performante, come ad esempio server dedicati. In questo modo si potrebbero sia utilizzare le tecniche di riduzione della dimensionalità che non riescono a convergere, sia allargare la griglia di iperparametri della *GridSearchCV*. Questo comporterebbe un aumento del costo computazionale ma si potrebbe riuscire a trovare un modello che si adatta meglio ai dati a disposizione e che riesca a ottenere un grado di bontà di generalizzazione superiore a quello ottenuto dai modelli che sono stati usati durante gli esperimenti. Un ulteriore sviluppo futuro possibile potrebbe essere quello di utilizzare algoritmi di apprendimento automatico più complessi, non solo *random forest*, come *artificial neural network* [100] o *XGBoost* [101–103].

Bibliografia

[1] George F Luger. *Artificial intelligence: structures and strategies for complex problem solving*. Pearson education, 2005.

[2] Tarun Lalwani, Shashank Bhalotia, Ashish Pal, Vasundhara Rathod, and Shreya Bisen. Implementation of a chatbot system using ai and nlp. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST) Volume-6, Issue-3*, 2018.

[3] Erik Cambria and Bebo White. Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57, 2014.

[4] Gaudenz Danuser. Computer vision in cell biology. *Cell*, 147(5):973–978, 2011.

[5] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[6] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997.

[7] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[8] Daniel Berrar. Cross-validation., 2019.

[9] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[10] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014.

[11] Kalyan Das, Jiming Jiang, and JNK Rao. Mean squared error of empirical predictor. 2004.

[12] A Colin Cameron and Frank AG Windmeijer. An r-squared measure of goodness of fit for some common nonlinear regression models. *Journal of econometrics*, 77(2):329–342, 1997.

[13] Carl Kingsford and Steven L. Salzberg. What are decision trees? *Nature Biotechnology*, 26(9):1011–1013, Sep 2008.

[14] Carl Kingsford and Steven L Salzberg. What are decision trees? *Nature biotechnology*, 26(9):1011–1013, 2008.

[15] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[16] Zoubin Ghahramani. *Unsupervised Learning*, pages 72–112. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[17] Laurence Morissette and Sylvain Chartier. The k-means clustering technique: General considerations and implementation in mathematica. *Tutorials in Quantitative Methods for Psychology*, 9(1):15–24, 2013.

[18] Eamonn Keogh and Abdullah Mueen. *Curse of Dimensionality*, pages 314–315. Springer US, Boston, MA, 2017.

[19] Luis O. Jimenez and David A. Landgrebe. Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *IEEE Trans. Syst. Man Cybern. Part C*, 28:39–54, 1998.

[20] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 07 2003.

[21] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 92–100, New York, NY, USA, 1998. Association for Computing Machinery.

[22] Qingyong Wang, Liang-Yong Xia, Hua Chai, and Yun Zhou. Semi-supervised learning with ensemble self-training for cancer classification. pages 796–803, 10 2018.

[23] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[24] Iqbal H. Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):160, Mar 2021.

[25] Pedro Larrañaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Iñaki Inza, José A. Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez, and Victor Robles. Machine learning in bioinformatics. *Briefings in Bioinformatics*, 7(1):86–112, 03 2006.

[26] `https://en.wikipedia.org/wiki/GenBank`.

[27] Catherine Mathé, Marie-France Sagot, Thomas Schiex, and Pierre Rouzé. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Res*, 30(19):4103–4117, October 2002.

[28] Stein Aerts, Peter Van Loo, Yves Moreau, and Bart De Moor. A genetic algorithm for the detection of new cis-regulatory modules in sets of coregulated genes. *Bioinformatics*, 20(12):1974–1976, March 2004.

[29] R J Carter, I Dubchak, and S R Holbrook. A computational approach to identify genes for functional RNAs in genomic sequences. *Nucleic Acids Res*, 29(19):3928–3938, October 2001.

[30] Atul Butte. The use and analysis of microarray data. *Nature reviews drug discovery*, 1(12):951–960, 2002.

[31] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[32] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[33] Max Schubach, Matteo Re, Peter N. Robinson, and Giorgio Valentini. Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants. *Scientific Reports*, 7(1):2959, Jun 2017.

[34] `https://ourworldindata.org/covid-deaths`.

[35] `https://ourworldindata.org/explorers/coronavirus-data-explorer`.

[36] Elena Casiraghi, Dario Malchiodi, Gabriella Trucco, Marco Frasca, Luca Cappelletti, Tommaso Fontana, Alessandro Andrea Esposito, Emanuele Avola, Alessandro Jachetti, Justin Reese, Alessandro Rizzi, Peter N Robinson, and Giorgio Valentini. Explainable machine learning for early assessment of COVID-19 risk prediction in emergency departments. *IEEE Access*, 8:196299–196325, October 2020.

[37] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 590–597, 2019.

[38] Jiechao Ma, Yang Song, Xi Tian, Yiting Hua, Rongguo Zhang, and Jianlin Wu. Survey on deep learning for pulmonary medical imaging. *Frontiers of medicine*, 14:450–469, 2020.

[39] Keno K Bressem, Lisa C Adams, Christoph Erxleben, Bernd Hamm, Stefan M Niehues, and Janis L Vahldiek. Comparing different deep learning architectures for classification of chest radiographs. *Scientific reports*, 10(1):13590, 2020.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[41] Ali Narin, Ceren Kaya, and Ziynet Pamuk. Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *Pattern Analysis and Applications*, 24:1207–1220, 2021.

[42] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[43] Dipayan Das, KC Santosh, and Umapada Pal. Truncated inception net: Covid-19 outbreak screening using chest x-rays. *Physical and engineering sciences in medicine*, 43:915–925, 2020.

[44] Ioannis D Apostolopoulos and Tzani A Mpesiana. Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. *Physical and engineering sciences in medicine*, 43:635–640, 2020.

[45] Kabid Hassan Shibly, Samrat Kumar Dey, Md Tahzib-Ul Islam, and Md Mahbubur Rahman. Covid faster r–cnn: A novel framework to diagnose novel coronavirus disease (covid-19) in x-ray images. *Informatics in Medicine Unlocked*, 20:100405, 2020.

[46] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67, 2011.

[47] Philipp Gaffert, Florian Meinfelder, and Volker Bosch. Towards an mi-proper predictive mean matching. In *Conf Proc*, 2016.

[48] Daniel J. Stekhoven and Peter Bühlmann. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 10 2011.

[49] Miron B. Kursa and Witold R. Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010.

[50] Miron B. Kursa, Aleksander Jankowski, and Witold R. Rudnicki. Boruta – a system for feature selection. *Fundamenta Informaticae*, 101:271–285, 2010. 4.

[51] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1):25, Jan 2007.

[52] Houtao Deng, George Runger, Eugene Tuv, and Wade Bannister. Cbc: An associative classifier with a small number of rules. *Decision Support Systems*, 59(1):163–170, March 2014. Funding Information: This research was partially supported by ONR grant N00014-09-1-0656 . Copyright: Copyright 2014 Elsevier B.V., All rights reserved.

[53] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384, 1972.

[54] Yehudit Hasin, Marcus Seldin, and Aldons Lusis. Multi-omics approaches to disease. *Genome Biology*, 18(1):83, May 2017.

[55] Rosalind Lee, Rhonda Feinbaum, Victor Ambros, et al. A short history of a short rna. *Cell*, 116(2 Suppl):S89–92, 2004.

[56] Thomas X Lu and Marc E Rothenberg. Microrna. *Journal of allergy and clinical immunology*, 141(4):1202–1207, 2018.

[57] Deniz M Ozata, Ildar Gainetdinov, Ansgar Zoch, Dónal O'Carroll, and Phillip D Zamore. Piwi-interacting rnas: small rnas with big functions. *Nature Reviews Genetics*, 20(2):89–108, 2019.

[58] Bryan R Cullen. Nuclear rna export. *Journal of cell science*, 116(4):587–597, 2003.

[59] Monica Sager, Nai Chien Yeat, Stefan Pajaro-Van der Stadt, Charlotte Lin, Qiuyin Ren, and Jimmy Lin. Transcriptomics in cancer diagnostics: developments in technology, clinical research and commercialization. *Expert Review of Molecular Diagnostics*, 15(12):1589–1603, 2015.

[60] Andreas Papassotiropoulos, Michael Fountoulakis, Travis Dunckley, Dietrich A Stephan, and Eric M Reiman. Genetics, transcriptomics, and proteomics of alzheimer's disease. *Journal of Clinical Psychiatry*, 67(4):652, 2006.

[61] Seow-Neng Chan, Eden Ngah Den Low, Raja Affendi Raja Ali, and Norfilza Mohd Mokhtar. Delineating inflammatory bowel disease through transcriptomic studies: current review of progress and evidence. *Intestinal research*, 16(3):374, 2018.

[62] Sarah L Stenton, Laura S Kremer, Robert Kopajtich, Christina Ludwig, and Holger Prokisch. The diagnosis of inborn errors of metabolism by an integrative "multi-omics" approach: A perspective encompassing genomics, transcriptomics, and proteomics. *Journal of inherited metabolic disease*, 43(1):25–35, 2020.

[63] `https://www.recentiprogressi.it/archivio/2296/articoli/24680/`.

[64] Zahra Momeni, Esmail Hassanzadeh, Mohammad Saniee Abadeh, and Riccardo Bellazzi. A survey on single and multi omics data mining methods in cancer data classification. *Journal of Biomedical Informatics*, 107:103466, 2020.

[65] `https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga`.

[66] Jianfang Liu et al. An integrated tcga pan-cancer clinical data resource to drive high-quality survival outcome analytics. *Cell*, 173(2):400–416.e11, 2018.

[67] Cristian Coarfa, Sandra L Grimm, Kimal Rajapakshe, Dimuthu Perera, Hsin-Yi Lu, Xuan Wang, Kurt R Christensen, Qianxing Mo, Dean P Edwards, and Shixia Huang. Reverse-Phase protein array: Technology, application, data processing, and integration. *J Biomol Tech*, 32(1):15–29, April 2021.

[68] Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10(1):57–63, January 2009.

[69] Anil Jadhav, Dhanya Pramod, and Krishnan Ramanathan. Comparison of performance of data imputation methods for numeric dataset. *Applied Artificial Intelligence*, 33(10):913–933, 2019.

[70] Akbar K Waljee, Ashin Mukherjee, Amit G Singal, Yiwei Zhang, Jeffrey Warren, Ulysses Balis, Jorge Marrero, Ji Zhu, and Peter DR Higgins. Comparison of imputation methods for missing laboratory data in medicine. *BMJ open*, 3(8):e002847, 2013.

[71] Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[72] Markus Ringnér. What is principal component analysis? *Nature biotechnology*, 26(3):303–304, 2008.

[73] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18(1998):1–8, 1998.

[74] Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. Singular value decomposition and principal component analysis. *A practical approach to microarray data analysis*, pages 91–109, 2003.

[75] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[76] `https://distill.pub/2016/misread-tsne/`.

[77] James M Joyce. Kullback-leibler divergence. In *International encyclopedia of statistical science*, pages 720–722. Springer, 2011.

[78] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016.

[79] `https://www.jmp.com/en_us/statistics-knowledge-portal/t-test/t-distribution.html`.

[80] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[81] Hervé Abdi. Singular value decomposition (svd) and generalized singular value decomposition. *Encyclopedia of measurement and statistics*, 907:912, 2007.

[82] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.

[83] `https://commons.wikimedia.org/wiki/File:Triangles_(spherical_geometry).jpg`.

[84] Vipin Kumar. Feature selection: A literature review. *The Smart Computing Review*, 4, 06 2014.

[85] Saptarsi Goswami and Amlan Chakrabarti. Feature selection: A practitioner view. *International Journal of Information Technology and Computer Science (IJITCS)*, 6(11):66, 2014.

[86] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. *Pearson Correlation Coefficient*, pages 1–4. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[87] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html`.

[88] Hanchuan Peng, Fuhui Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and

min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[89] Witold R. Rudnicki, Marcin Kierczak, Jacek Koronacki, and Jan Komorowski. A statistical method for determining importance of variables in an information system. In Salvatore Greco, Yutaka Hata, Shoji Hirano, Masahiro Inuiguchi, Sadaaki Miyamoto, Hung Son Nguyen, and Roman Słowiński, editors, *Rough Sets and Current Trends in Computing*, pages 557–566, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[90] Y Hu, T Palmé, and O Fink. Maximal information-based nonparametric exploration of condition monitoring data". *PHM Society European Conference*, 3, 2016.

[91] David Reshef, Yakir Reshef, Hilary Finucane, Sharon Grossman, Gilean McVean, Peter Turnbaugh, Eric Lander, Michael Mitzenmacher, and Pardis Sabeti. Detecting novel associations in large data sets. *Science (New York, N.Y.)*, 334:1518–24, 12 2011.

[92] Robert S. Bennett. The intrinsic dimensionality of signal collections. *IEEE Trans. Inf. Theory*, 15:517–525, 1969.

[93] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Sci Rep*, 7(1):12140, September 2017.

[94] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`.

[95] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`.

[96] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[97] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):1–21, 03 2015.

[98] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[99] `https://www.intel.it/content/www/it/it/products/sku/230493/intel-core-i513600k-processor-24m-cache-up-to-5-10-ghz/specifications.html`.

[100] Paulo J Lisboa and Azzam FG Taktak. The use of artificial neural networks in decision support in cancer: a systematic review. *Neural networks*, 19(4):408–415, 2006.

[101] Xin Yu Liew, Nazia Hameed, and Jeremie Clos. An investigation of xgboost-based algorithm for breast cancer classification. *Machine Learning with Applications*, 6:100154, 2021.

[102] Sajib Kabiraj, M Raihan, Nasif Alvi, Marina Afrin, Laboni Akter, Shawmi Akhter Sohagi, and Etu Podder. Breast cancer risk prediction using xgboost and random forest algorithm. In *2020 11th international conference on computing, communication and networking technologies (ICCCNT)*, pages 1–4. IEEE, 2020.

[103] Yulin Zhang, Tong Feng, Shudong Wang, Ruyi Dong, Jialiang Yang, Jionglong Su, and Bo Wang. A novel xgboost method to identify cancer tissue-of-origin based on copy number variations. *Frontiers in genetics*, 11:585029, 2020.

[104] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022.