UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica

Elaborato finale in **Elaborazioni di Segnali Multimediali**

Classificazione di immagini biomedicali mediante descrittori locali

Anno Accademico 2019/2020

Candidato:

Raffaele Pane

N46003967

A chi c'è stato, ed anche a chi
ha deciso di non esserci.
Ma soprattutto, a chi è stato
sempre un buon motivo per non
accontentarsi.

Indice

Introduzione

È da ormai diversi anni che si riscontra un notevole interesse da parte della comunità scientifica verso l'applicazione di tecniche di classificazione di immagini in ambito biomedicale. Risulta evidente come esse permettano eccellenti performance, soprattutto se confrontate con le più tradizionali tecniche di diagnosi e prognosi di malattie.

Un esempio classico è quello del riconoscimento e della classificazione delle microcalcificazioni al seno.

Il tumore al seno, nonostante la grande presa di coscienza del problema da parte dell'opinione pubblica e della ricerca scientifica, continua ad essere una delle maggiori cause di morte da cancro, soprattutto nella popolazione femminile. Prendendo i dati presenti nel report redatto da AIOM (Associazione Italiana di Oncologia Medica) per il 2020 [1] a circa una donna su otto può essere diagnosticato il cancro alla mammella lungo il corso della propria vita.

Come sempre in questi casi, un rilevamento nelle prime fasi del processo di calcificazione unita ad una diagnosi tempestiva ed accurata, risultano fondamentali per aumentare quelle che sono le possibilità di sopravvivenza a lungo termine del paziente.

Sono quindi necessarie delle tecniche di diagnostica rapide, precise ed affidabili, che possano supportare al meglio il lavoro del medico.

Una possibilità è quella di utilizzare gli algoritmi di Machine Learning per la predizione di una diagnosi, partendo da descrittori (feature) ricavati dai nuclei di cellule tumorali

estratte tramite FNA (Fine Needle Aspiration). Si tratta di una metodologia poco invasiva, ad elevata accuratezza (sempre al di sopra del 90%) e con poca varianza nei risultati, proponendosi quindi come una soluzione concreta al suddetto problema.

In questo elaborato si è partiti dal famoso Breast Cancer Wisconsin (Diagnostic) Data Set [2], a cui sono stati applicati vari algoritmi di classificazione tramite apprendimento supervisionato. Si sono poi valutati accuratezza e precisione dei vari algoritmi, così da poter dare un quadro completo delle capacità e delle potenzialità di ognuno di essi.

.

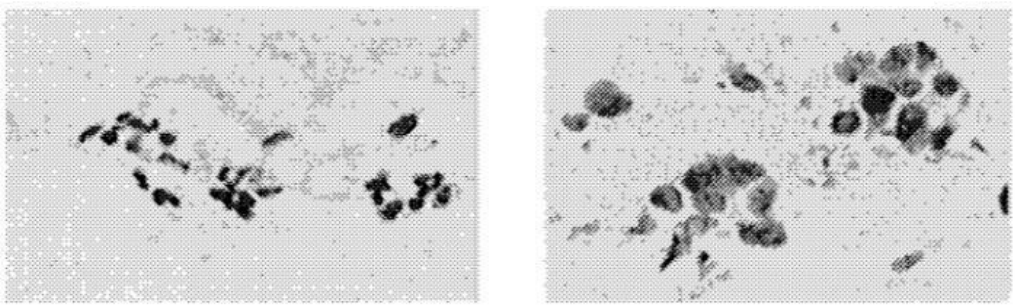
Capitolo 1: Problema

La maggior parte dei tumori al seno sono rilevati dal paziente sotto forma di nodulo, a cui fa poi seguito un necessario approfondimento medico per carpirne la natura benigna o maligna. Le principali tecniche di diagnosi sono quelle della mammografia, dell'agoaspirazione (FNA) con interpretazione visiva e della biopsia chirurgica.

Dalle statistiche riscontrate però risulta che la precisione con cui vengono correttamente diagnosticati i tumori con tali tecniche varia dal 68% al 79% [3] per la mammografia, dal 65% al 98% [4] per l'agoaspirazione con interpretazione visiva ed è quasi del 100% per la biopsia chirurgica. Risulta quindi evidente come la mammografia sia una tecnica non molto precisa, l'agoaspirazione con interpretazione visiva soffra invece di una varianza molto alta, mentre la biopsia chirurgica, benché accurata, sia una pratica invasiva, lunga e costosa.

Unire quindi i vantaggi dati dall'agoaspirazione a quelli della classificazione tramite descrittori locali risulta fondamentale, in quanto permette di avere sia un'alta accuratezza nella diagnosi che una pratica in generale poco invasiva e con tempi di risposta brevi.

Figura 1: Immagini digitalizzate di FNA [2][10]: a sinistra Benigno, a destra Maligno



Capitolo 2: Metodologia

Esistono svariate tecniche per la classificazione delle immagini. Una tra le più innovative e fruttuose è quella che propone l'utilizzo degli algoritmi del Machine Learning per la produzione di un modello predittivo in grado di classificare in maniera affidabile le immagini poste in ingresso.

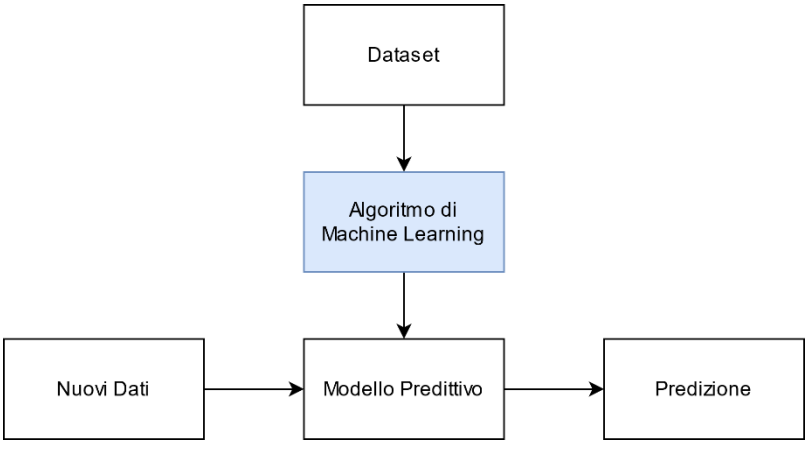
Ma cos'è il Machine Learning?

L'apprendimento Automatico (o Machine Learning) è una delle tecniche di Intelligenza Artificiale più interessanti del nostro tempo.

Essa si propone di "insegnare" alle macchine a fare ciò che uomo ed animali già fanno naturalmente: imparare dall'esperienza.

Gli algoritmi del Machine Learning usano dei metodi computazionali per "apprendere" informazioni utili direttamente dai dati in maniera autonoma, senza l'uso di un'equazione predeterminata come modello e quindi di istruzioni esplicite.

Figura 2: Grafico riassuntivo del procedimento



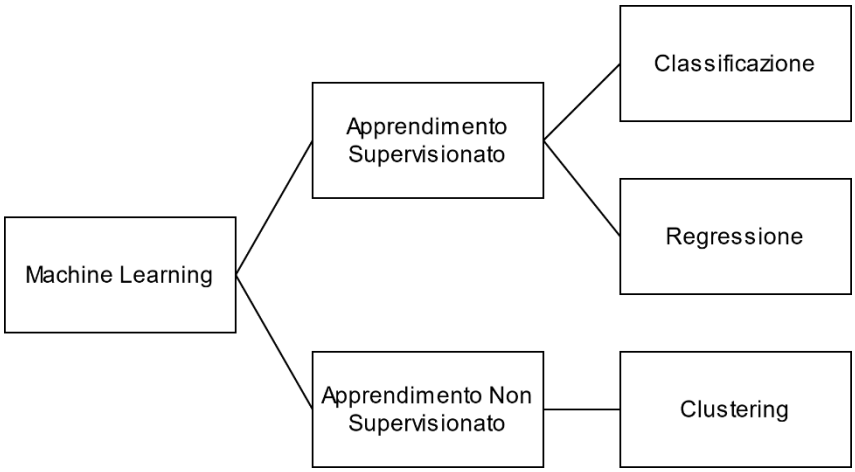


Figura 3:Schema sulle varie tipologie di apprendimento automatico

I compiti dell'apprendimento automatico vengono tipicamente classificati in tre ampie categorie, a seconda dei tipi di feedback che possono accompagnare gli input, e che determinano i tre tipi principali di apprendimento [5][6]:

- apprendimento supervisionato, in cui viene sviluppato un modello predittivo basandosi sia sui dati di input che su quelli di output; all'agente vengono forniti degli esempi nella forma di possibili input e i rispettivi output desiderati, e l'obiettivo è quello di estrarre una regola generale che associ l'input all'output corretto (es: Classificazione, Regressione);
- apprendimento non supervisionato, in cui il modello ha lo scopo di trovare una struttura negli input forniti, senza che gli input vengano etichettati in alcun modo (es: Clustering);
- apprendimento per rinforzo, in cui il modello interagisce con un ambiente dinamico nel quale cerca di raggiungere un obiettivo, avendo un insegnante che gli dice solo se ha raggiunto l'obiettivo; l'agente impara quindi da una serie di rinforzi: premi e punizioni.

L'apprendimento supervisionato può essere ulteriormente categorizzato tenendo in considerazione l'output desiderato del sistema:

- nella classificazione, gli output sono divisi in due o più classi e il sistema di

apprendimento deve produrre un modello che assegni gli input non ancora visti a una o più di queste; risulta quindi una predizione di tipo discreto;

- nella regressione, invece, l'output e il modello utilizzati sono continui.

Per il nostro caso risulta quindi evidente che avremo bisogno di un Classificatore, e quindi di un Apprendimento Supervisionato.

Il problema che ci si pone adesso dinanzi è scegliere quale fra i tanti algoritmi di classificazione disponibili sia l'ideale per il nostro compito. Risulta evidente che ogni algoritmo ha le sue peculiarità e si basa su specifici presupposti, rendendo quindi tale operazione per nulla banale.
Oltre che una certa dimestichezza ed esperienza, risulta quindi consigliabile confrontare direttamente le prestazioni di almeno un ristretto gruppo di algoritmi di apprendimento, così da poter effettuare la scelta migliore per lo specifico problema.
Infatti, gli algoritmi lavorano sui dati, e quindi la natura e le dimensioni del dataset influenzano enormemente la qualità dei modelli. Parametri come il numero di caratteristiche, il livello di rumore o la dispersione nello spazio delle varie classi (e quindi quanto sia facile poi separarle: lineare, quadratica, ecc) sono decisivi per una corretta predizione.

Dobbiamo quindi implementare vari tipi di algoritmi di classificazione e poi testarne le performance. Per dare ciò però abbiamo bisogno di una comune metrica prestazionale. Ne esistono svariate, ma in questo elaborato si è scelta l'accuratezza, definita come:

$$Accuratezza = \frac{n° \; di \; predizioni \; corrette}{N° \; totale \; di \; predizioni \; fatte}$$

Tale metrica andrà valutata su dei dati che devono per forza di cose essere diversi da quelli usati per addestrare il modello, altrimenti la precisione verrebbe sempre garantita (commettendo un banale errore logico). Sorge allora l'esigenza di dividere il dataset in

due parti: una per l'addestramento (training) ed una per la valutazione (test).
In letteratura sono presenti le più svariate percentuali; in questo caso si è proceduto con una suddivisione del 80% per la fase di training e del 20% per quella di test, presi casualmente dal dataset di partenza.
Si impiega il dataset di addestramento per addestrare e ottimizzare il proprio modello di machine learning, mentre si mette da parte fino alla fine il dataset di test, per valutarne poi il modello finale.
Abbiamo quindi tutti i presupposti concettuali per poter procedere con l'implementazione del modello.

In generali è possibile sintetizzare i passi coinvolti nell'addestramento di un algoritmo di Machine Learning con Supervisione come segue:

1) preparazione del dataset;
2) addestramento di un modello, usando il dataset di training elaborato ed uno degli algoritmi;
3) testing del modello sul dataset di test.

Tale processo ovviamente andrà iterato sia sul singolo modello, così da ottimizzarlo attraverso il tuning dei parametri del classificatore, che globalmente per provare algoritmi differenti, e quindi poter effettuare una comparazione.

2.1 Dataset

Come già detto in precedenza, si è utilizzato il Wisconsin Diagnostic Breast Cancer per l'implementazione degli algoritmi di Machine Learning per la classificazione.

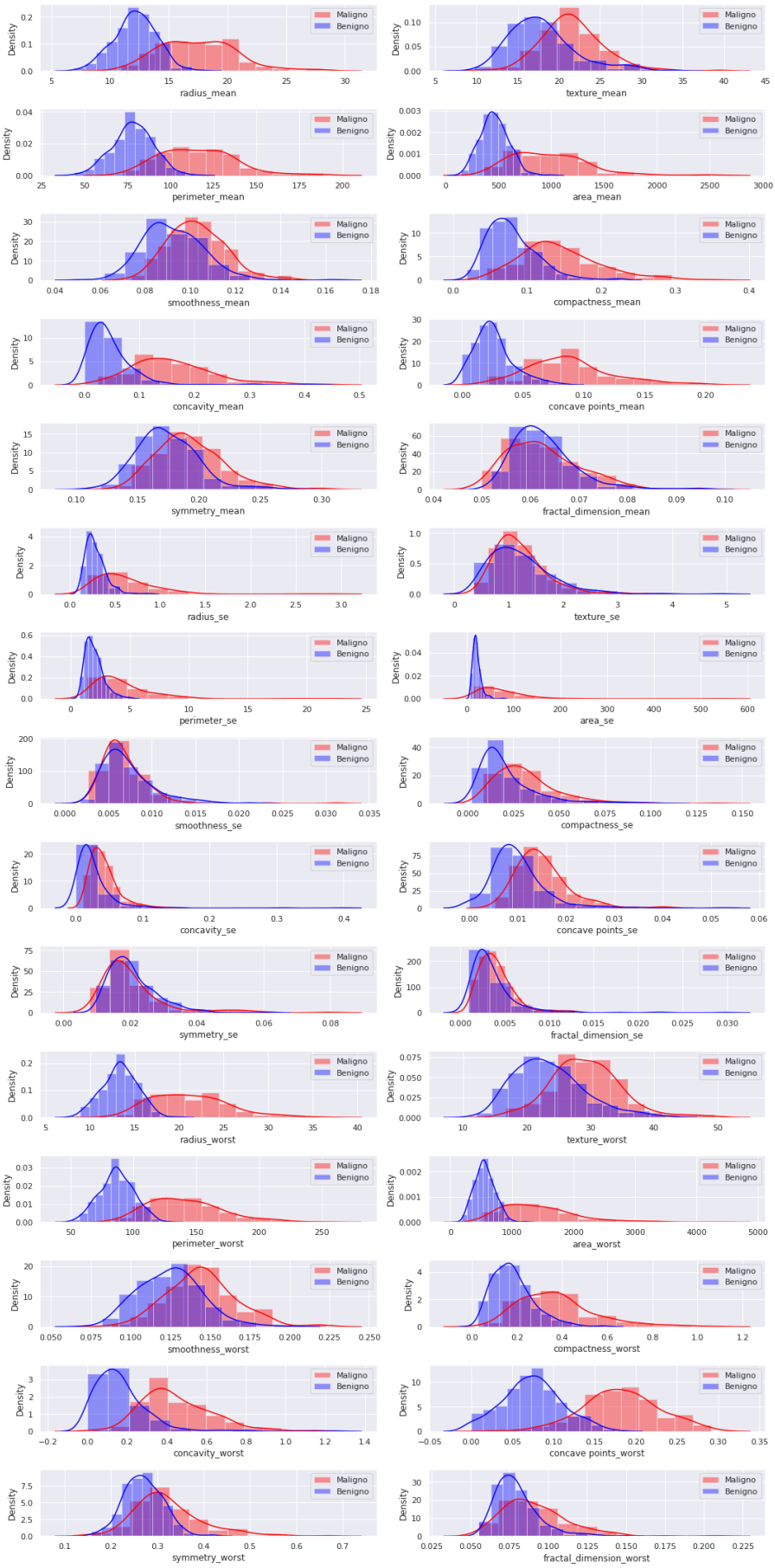
Il dataset è composto da 569 istanze, di cui 212 sono etichettate come Maligne ed altre 357 come Benigne. Ogni istanza è caratterizzata da 30 attributi (feature) derivanti dai nuclei cellulari presi da immagini digitalizzate di masse tumorali estratte con tecnica FNA. La metodologia di estrazione di tali descrittori è descritta dettagliatamente in [7] e fa uso della tecnica dello snake, così da permette un'analisi precisa e automatizzata delle dimensioni, della forma e della struttura dei nuclei. Per ogni nucleo sono calcolate dieci caratteristiche:

- radius (media delle distanze dal centro ai vari punti del perimetro)
- texture (deviazione standard dei valori nella scala dei grigi)
- perimeter
- area
- smoothness
- compactness (perimeter^2 / area - 1.0)
- concavity
- concave points (numero di porzioni concave del bordo)
- symmetry
- fractal dimension ("coastline approximation" - 1)

di cui poi vengono calcolati, per ogni istanza, i valori medi, il valore più grande (o "peggiore") e l'errore standard, per un totale di 30 feature.

Di seguito è stata inserita una visualizzazione dei valori delle singole feature presenti all'interno del dataset. Per rappresentarle si è utilizzata la funzione *distplot()* della libreria *seaborn*. Si può notare come per alcune di esse i valori etichettati siano abbastanza ben localizzati nello spazio, e possono quindi suggerire una selezione delle caratteristiche da utilizzare poi nell'implementazione.

Figura 4:Visualizzazione delle singole caratteristiche del dataset, colorate a seconda dell'etichetta



2.2 Pre-Elaborazione dei Dati

In generale i dati possono non presentarsi nella forma e nella scala ideale per permettere ai vari algoritmi di lavorare al meglio. Ad esempio, si può notare come le varie feature dell'immagine 2, sono in scale differenti.

Per evitare che vi siano problemi di assegnamento dei valori di rilevanza, si standardizza il dataset tramite *StandardScaler()*:

$$z = \frac{(X - \mu)}{\sigma}$$

Dove X è la feature da standardizzare, μ è la media e σ è la deviazione standard.

In questo modo tutti i valori saranno compresi nell'intervallo [0, 1], a media nulla e a varianza unitaria.

Un ulteriore problema che può presentarsi è che le caratteristiche possono essere fra loro fortemente correlate, e pertanto forniscono informazioni ridondanti. In questi casi, le tecniche di riduzione della dimensionalità possono essere molto utili per la compressione delle caratteristiche in un sottospazio di dimensionalità inferiore. Ciò offre il duplice vantaggio di ridurre sia i tempi di esecuzione dell'algoritmo di apprendimento che lo spazio di memoria richiesto.

Inoltre, in alcuni casi, la riduzione della dimensionalità può anche migliorare le prestazioni predittive di un modello (ad esempio in presenza di rumore).

Una di queste tecniche è la Principal Component Analysis (PCA). Essa punta a trovare le direzioni di massima varianza all'interno dei dati di partenza, per poi farli diventare gli assi ortogonali di un nuovo spazio, a dimensionalità minore, su cui saranno proiettati i dati. È quindi un vero e proprio cambio di coordinate, una nuova rappresentazione del dato stesso che, come è facile intuire, non comporta alcuna perdita di informazione.

In Python è facilmente implementabile tramite l'omonima *PCA*, in cui è anche possibile settare il numero di componenti che formeranno la rappresentazione d'uscita *(n_components=20).*

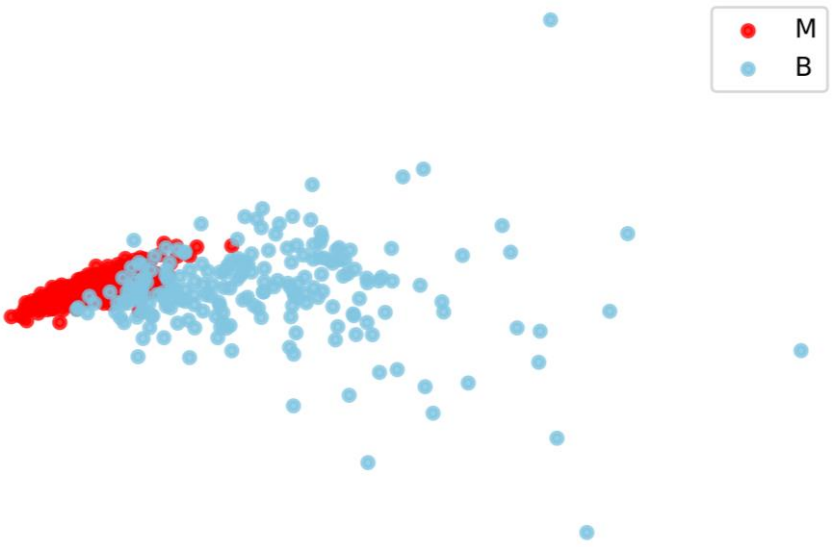


Figura 5: Visualizzazione della PCA del dataset per n° di componenti uguale a 2

2.3 Algoritmi di Machine Learning

Come detto in precedenza, esistono decine di algoritmi per l'apprendimento automatico, ognuno con le proprie caratteristiche e qualità.

In questo elaborato sono state implementati e confrontati alcuni fra i più famosi classificatori, tutti presenti nella libreria *scikit-learn* [8] di Python.

Sono poi state usate altre librerie come *matplotlib*, *pandas*, *seaborn*, *numpy* e *time* per la creazione di grafici, l'analisi e la raccolta dei risultati.

2.3.1 Nearest Neighbors

Il principio alla base dei metodi Nearest Neighbors è quello di trovare, partendo da un punto target da classificare, un numero predefinito k (anche se esiste una ulteriore versione che utilizza un valore k variabile) di campioni presi per vicinanza (neighborhood).

Sulla base della metrica per la distanza scelta, l'algoritmo K-Nearest Neighbors trova i k

valori presenti nel dataset di addestramento che sono più prossimi nello spazio al punto che vogliamo classificare. L'etichetta della classe del punto viene poi determinata in base alla categoria con maggior occorrenza tra i k valori presi.

Metriche tipiche per la distanza sono quella Euclidea ($p = 2$) o quella Manhattan ($p = 1$), entrambe casi particolari della più generale distanza di Minkowski:

$$\left(\sum_{i=1}^{n} |x_i - z_i|^p\right)^{1/p}$$

dove x_i e z_i sono le i-sime componenti dei punti X e Z.

In questo elaborato si è usato *KNeighborsClassifier()*, sia con metrica Manhattan che Euclidea, senza tuttavia riscontrare differenze prestazionali sostanziali. Si è inoltre scelto come numero di neighbors $k = 5$.

2.3.2 Naive Bayes

I Naive Bayes sono una famiglia di classificatori probabilistici basati sull'applicazione del teorema di Bayes nell' ipotesi di forte indipendenza tra le caratteristiche (feature) di partenza. Da qui l'aggettivo "naïve", in quanto tale ipotesi viene spesso nei fatti violata, mantenendo comunque delle performance molto buone (nonostante quindi l'irrealistico assunto). [9]

È quindi relativamente robusto, semplice da implementare, rapido ed accurato.

Il teorema di Bayes può essere scritto come di seguito:

$$P(y\,|x_i, \dots, x_n) = \frac{P(y)P(x_i, \dots, x_n\,|y)}{P(x_i, \dots, x_n)}$$

dove $P(y\,|x_i, \dots, x_n)$ è la probabilità dell'evento y condizionata dagli eventi $x_i, \dots, x_n$.

Si è proceduto all'implementazione di due varianti della famiglia dei Naive Bayes, diversificati dall'assunto che si fa sulla forma della distribuzione di probabilità delle caratteristiche:

- Gaussiana, utilizzando la classe *GaussianNB()*

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma_y^2}\right)$$

- di Bernoulli, attraverso *BernoulliNB()*, che fa uso della seguente formula:

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i)$$

2.3.3 Support Vector Machine (SVM)

Il Support Vector Machine (SVM) è un algoritmo di Machine Learning supervisionato particolarmente efficace nei problemi di classificazione binaria.

L'SVM è basato sull'idea di trovare un iperpiano separatore che divida al meglio un set di dati in due classi. Se ne esiste più di uno, cerca quello che ha margine (distanza tra i vettori di supporto di classi differenti più vicini all'iperpiano) maggiore.

Se tale iperpiano non esiste, SVM realizza una mappatura non lineare per aumentare di dimensione lo spazio dei dati di training, così da poter effettuare la separazione. Questo metodo si basa sull'idea che problemi non separabili in uno spazio ad n dimensioni, lo diventino in un generico spazio ad almeno n+1 dimensioni.

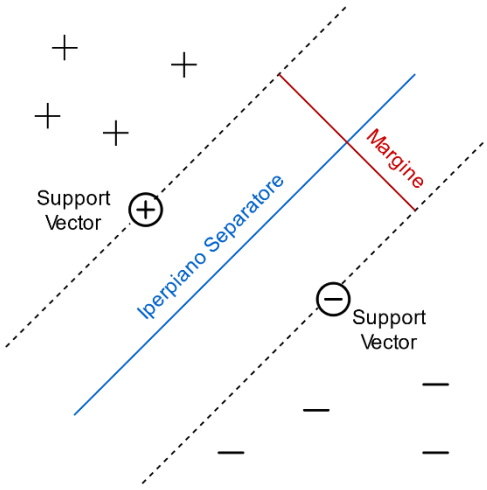


Figura 6: Esempio di separazione lineare, usando le SVM

Si sono implementate e valutate due versioni di SVM, usando una volta un kernel lineare ed un'altra uno RBF (Radial Basis Function), entrambi parametri della classe *SVC()*.

2.3.4 Decision Tree

Un Albero di Decisione (Decision Tree) è una semplice rappresentazione, dalla tipica forma ad albero capovolto, formata da nodi collegati tra loro da linee. In ogni nodo dell'albero c'è un fattore (domanda) in base al quale si prosegue per uno dei rami (risposte) sottostanti, fino ad arrivare ad una decisione. Si parte quindi dal nodo radice (root) e si prosegue per scelte successive fino a raggiungere uno dei nodi foglia del grafo.
Nel caso di un classificatore la struttura dell'albero di decisione viene costruita apprendendo dalle informazioni estratte dal dataset di training, mentre la decisione finale è l'etichetta da applicare come predizione fatta.

In Python è possibile usare un Classificatore basato su Albero di Decisione attraverso la classe *DecisionTreeClassifier()*.
Il maggior vantaggio di questa tecnica risulta quindi essere la semplice interpretabilità del modello, grazie anche a tool grafici che ne permettono una facile rappresentazione, mentre

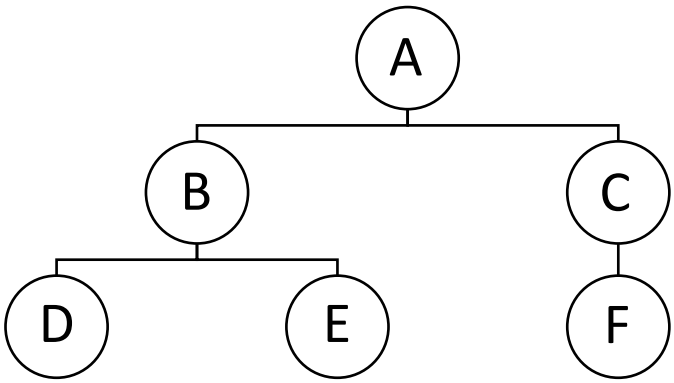


Figura 7: Esempio di Albero Decisionale

lo svantaggio maggiore risulta quello che il classificatore può tendere a creare alberi troppo complessi che non generalizzano bene i dati (fenomeno dell'overfitting). Meccanismi come la potatura (pruning) o l'impostazione della profondità massima che può avere un albero, sono necessari per evitare questo problema.

2.3.5 Random Forest

La tecnica Random Forest consiste nell'utilizzare contemporaneamente diversi alberi di decisione: si userà poi come decisione finale quella scelta dalla maggioranza degli alberi. Questa tecnica consente quindi di avere diversi punti di vista (alberi decisionali) sullo stesso problema di classificazione, comportando spesso risultati migliori. Tuttavia, si perde la capacità di generare in output quelle regole che rendono gli alberi decisionali uno strumento di facile interpretazione.

È stato implementato tramite la classe *RandomForestClassifier()*, popolando la foresta con 100 alberi decisionali.

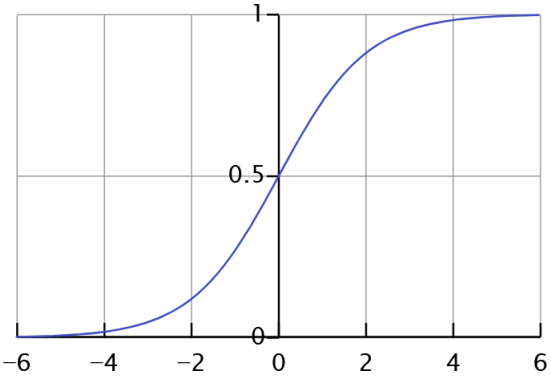


Figura 8: Curva logistica (sigmoide)

2.3.6 Logistic Regression

La Regressione Logistica, nonostante il nome, è un modello lineare di classificazione binaria, rientrante nella famiglia degli algoritmi di apprendimento supervisionato.

Avvalendosi di metodi statistici, permette di generare un risultato che, di fatto, rappresenta una probabilità che un dato valore di ingresso appartenga ad una determinata classe.

Tale probabilità è modellata utilizzando una funzione logistica, ovvero una curva di crescita a forma di S, che preso in ingresso valori reali restituisce un valore compreso tra 0 e 1 (dove 0 rappresenta una classe ed 1 l'altra).

L'espressione generale della funzione Logistica è:

$$P(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

dove x_0 è il valore del punto medio del sigmoide, L il suo valore massimo e k la sua ripidità.

La Regressione Logistica è implementata tramite *LogisticRegression*().

Nel caso multiclasse, l'algoritmo di addestramento utilizza lo schema one-vs-rest (OvR) oppure la funzione Softmax (è una generalizzazione multidimensionale della funzione logistica), a seconda se il parametro '*multi_class*' sia settato rispettivamente su '*ovr*' o su '*multinomial*'.

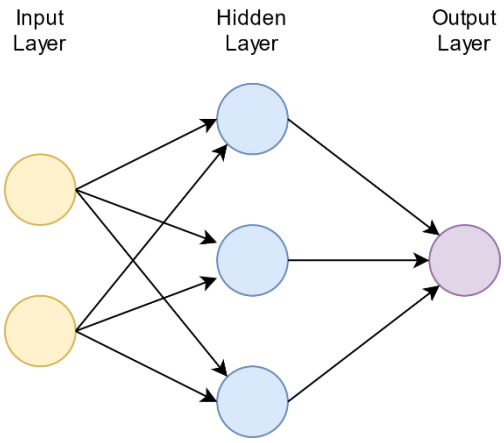


Figura 9: Esempio di MLP

2.3.7 Multi-Layer Perceptron

Per poter descrivere al meglio il Multi-Layer Perceptron (MLP) bisogna prima introdurre il concetto di Rete Naturale Artificiale (ANN).

Le Reti Neurali Artificiali sono modelli che provano a riprodurre il funzionamento delle Reti Neurali Biologiche del cervello animale, relativamente al modo con cui risolve problemi complessi. Una ANN è formata da una collezione di nodi (anche detti Neuroni Artificiali) connessi fra loro da rami (le Sinapsi), tali quindi da formare una rete. Proprio come in un cervello biologico ogni neurone riceve un segnale attraverso le sinapsi, lo elabora, per poi inviarlo a sua volta agli altri nodi a lui collegati.

Possiamo allora definire il Multi-Layer Perceptron come una classe delle Reti Neurali Artificiali Feed-Foward, dove la connessione tra i nodi è direzionale ("in avanti") e non forma cicli. Inoltre, negli MLP i neuroni sono dislocati in livelli (Layer), dove ogni livello ha connessioni entranti dal precedente strato ed uscenti in quello successivo.

I livelli possono essere di tre tipologie:

1) Input Layer: riceve le informazioni provenienti dall'esterno;
2) Output Layer: livello finale che mostra il risultato di quanto il programma è riuscito ad imparare;

3) Hidden Layer: collegano il livello di ingresso con quello di uscita ed aiutano la rete neurale ad apprendere le relazioni complesse analizzate dai dati. Spesso i livelli nascosti sono più di uno.

Eccezion fatta che per i nodi in ingresso, ogni nodo possiede una funzione di attivazione non lineare attraverso la quale, dato un input, si definisce l'uscita. Esempi tipici sono la funzione logistica, la tangente iperbolica o la più recente ReLU(Rectifier Linear Unit).

Per questa implementazione si è utilizzata la classe *MLPClassifier()*, con funzione di attivazione ReLu. La struttura è formata da 3 Hidden Layer da 500 nodi ognuno.

Capitolo 3: Risultati

3.1 Valutazione

La necessità di valutare un modello viene data dal fatto che vi può essere un sostanziale distacco tra ciò che sono i dati di addestramento rispetto a quelli che si avranno poi per effettuare le predizioni effettive. È quindi necessario testare quelle che sono le sue capacità di generalizzazione, la flessibilità con cui si adatta al rispondere a dati mai visti.
Risulta evidente che quindi sia necessario testare le performance del modello su un set di dati diverso da quello di addestramento, ad esempio riservando una parte del dataset di partenza per tale fase. Ciò risulta essere particolarmente semplice nel caso di Apprendimento Supervisionato, in quanto il dataset di partenza è già per sua natura etichettato. Si può quindi procedere ad un semplice confronto diretto tra le etichette di partenza e quelle assegnate dal classificatore a seguito della predizione.

Partendo da questi presupposti si possono ricavare due semplici metriche per la valutazione: l'accuratezza, definita come il rapporto tra le predizioni esatte su quelle totali, e la Matrice di Confusione. Quest'ultima è una matrice le cui righe rappresentano i valori reali, mentre le colonne quelli predetti. Saranno allora presenti lungo la diagonale principale le predizioni corrette, mentre nelle restanti posizioni rispettivamente i casi Falsi Positivi (FP) e Falsi Negativi (FN).

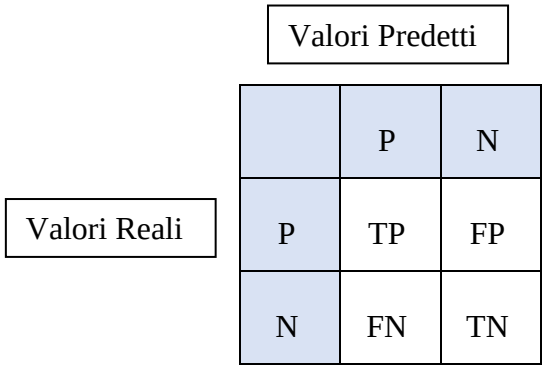


Figura 10: Schema di una Matrice di Confusione

Dal momento che si cerca comunque di ottimizzare e confrontare i parametri impostabili di ogni singolo modello, così da raggiungere dei valori ottimali, sorge allora un' ulteriore problematica. Infatti, se si usasse il dataset di test per validare i vari parametri, li si farebbe di fatto rientrare nel dataset di addestramento, comportando un maggior rischio di overfitting.

L'overfitting è quel fenomeno nel quale il modello si adatta eccessivamente alle caratteristiche specifiche del training set, perdendo quindi l'abilità di generalizzazione. È un fenomeno che solitamente si presenta soprattutto nei casi in cui l'apprendimento è stato effettuato troppo a lungo o se l'algoritmo è troppo complesso o flessibile.

Una prima soluzione è quella dell'Holdout, in cui si "trattiene" una parte del set di addestramento per effettuare la convalida. Tale soluzione offre però lo svantaggio di partizionare ulteriormente il training set, esponendo la stima ad un rischio di eccessiva sensibilità allo specifico set di convalida.

Una soluzione migliore, utilizzato anche in questo lavoro, è quella della Convalida Incrociata K-fold. In questo metodo si divide il set di addestramento in k fold (pieghe) tra loro disgiunte, di cui una viene usata per la valutazione e le altre k-1 per l'addestramento. Tale procedura viene poi iterata per k volte, cambiando di volta in volta il fold di valutazione, così che ogni fold è usato esattamente una volta per la valutazione mentre le altre k-1 volte sarà stato parte del training.

Ad ogni iterazione verrà quindi calcolata la specifica metrica che si è utilizzata per

misurare le performance del modello (accuratezza, errore, ecc). Si farà poi una media di tali k valori così da avere un parametro prestazionale medio che mostra chiaramente quanto il modello è effettivamente in grado di generalizzare.
Come dimostrato da [11] un buon numero di fold con cui poter effettuare la Convalida Incrociata è 10.

3.2 Discussione

Tutti gli esperimenti di questo studio sono stati condotti sulla piattaforma Google Colab, provvista di risorsa Backend Google Compute Engine Python 3 e con runtime impostato senza accelerazione hardware.

Nella tabella 1 sono riassunti i risultati per ognuno degli algoritmi utilizzati, i cui parametri sono stati impostati manualmente. Come è evidente da una rapida visione, ogni algoritmo ha espresso ottime performance sia in termini di accuratezza riscontrata che nella capacità di generalizzazione (si veda il Cross-Validation Score). Inoltre, anche i tempi di esecuzione sono, ad eccezione dell'MLP, tutti al di sotto del secondo. Per valutarli si è usata la classe *time*, cronometrando esclusivamente la durata temporale della pipeline formata da pre-elaborazione, addestramento e predizione.
Come specificato in precedenza, si è diviso il dataset in due porzioni: 80% per la fase di addestramento, 20% per la fase di testing. Pertanto, i risultati saranno strettamente dipendenti da tale scelta, e sarebbero cambiati considerevolmente al variare dei rapporti scelti.

	Accuratezza	Cross-Validation Score	Tempo di Esecuzione
NEAREST NEIGHBORS	96.49	96.94 +/- 2.95	0.011090
NAYVE BAYES Gaussian	95.61	92.77 +/- 3.50	0.002349
NAYVE BAYES Bernoulli	94.74	93.41 +/- 4.59	0.002589
SVM Linear	96.49	97.59 +/- 2.06	0.005345
SVM rbf	97.37	97.80 +/- 2.41	0.008036
DECISION TREE	92.98	94.08 +/- 3.54	0.017686
RANDOM FOREST	94.74	96.48 +/- 1.76	0.281132
LOGISTIC REGRESSION	98.25	98.02 +/- 2.06	0.015164
MLP	98.25	97.59 +/- 1.52	33.637150

Tabella 1: Sommario dei risultati ottenuti

	Matrice di Confusione	TP	FP	FP/TotP%	TN	FN	FN/TotN%
NEAREST NEIGHBORS	[[71, 1], [3, 39]]	71	1	1.39 %	39	3	7.14 %
NAYVE BAYES Gaussian	[[71, 1], [4, 38]]	71	1	1.39 %	38	4	9.52 %
NAYVE BAYES Bernoulli	[[69, 3], [3, 39]]	69	3	4.17%	39	3	7.14 %
SVM Linear	[[70, 2], [2, 40]]	70	2	2.78 %	40	2	4.76 %
SVM rbf	[[71, 1], [2, 40]]	71	1	1.39 %	40	2	4.76 %
DECISION TREE	[[68, 4], [4, 38]]	68	4	5.56%	38	4	9.52 %
RANDOM FOREST	[[71, 1], [5, 37]]	71	1	1.39 %	37	5	11.9 %
LOGISTIC REGRESSION	[[71, 1], [1, 41]]	71	1	1.39 %	41	1	2.38 %
MLP	[[71, 1], [1, 41]]	71	1	1.39 %	41	1	2.38 %

Tabella 2: Riassunto delle Matrici di Confusione ottenute

Gli algoritmi che meglio hanno performato nei test effettuati sono quelli della Regressione Logistica e dell'MLP. Entrambi, infatti, hanno dei valori di accuratezza superiori al 98%, con altrettanto buoni test di validazione incrociata (si noti che oltre all'eccellente valore medio, anche lo scostamento massimo dei risultati è di massimo due punti percentili). Risulta però, tra i due, preferibile la Regressione Logistica in quanto richiede dei tempi di esecuzione, oltre che delle risorse, nettamente inferiori.

Passando invece ad algoritmi leggermente meno performanti, si può notare che quelli che soffrono maggiormente sono soprattutto il Decision Tree e il Nayve Bayes con distribuzione di probabilità di Bernoulli. In entrambi i casi, più che il valore dell'accuratezza, preoccupa quello della convalida incrociata. Infatti, si nota un valore

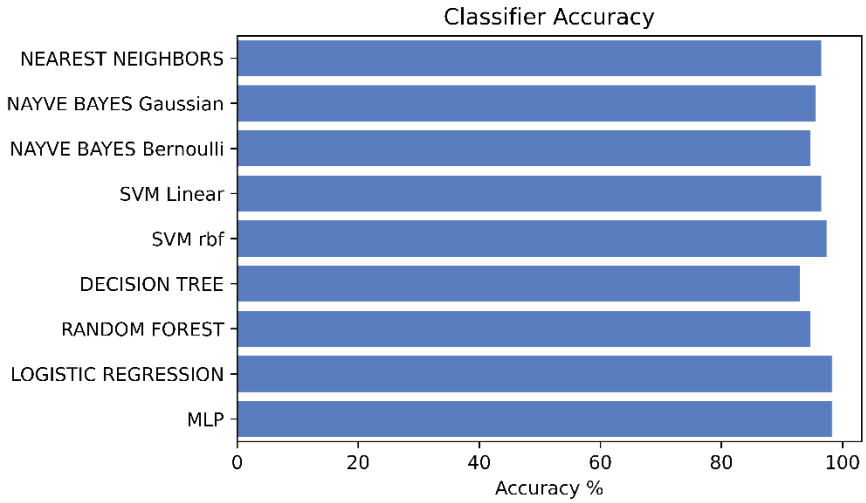


Figura 11: Ulteriore rappresentazione dei risultati

medio non entusiasmante, nonché dei valori di varianza superiori al 3.5%; a dimostrazione di una probabile bassa capacità di generalizzazione del modello. In entrambi i casi quindi si dovrebbe procedere ad un tuning migliore dei parametri.

Un altro aspetto particolarmente interessante è come le performance migliorino di un paio di punti percentili se si passa dall'algoritmo Decision Tree a quello Random Forest. È evidente allora il vantaggio che offre quest'ultimo nell'avere quindi un sistema di polling interno che gli permetta di superare le debolezze del singolo albero decisionale. Tutto ciò a svantaggio di una perdita di rappresentabilità tipica del Decision Tree e di tempi di calcolo leggermente più lunghi, sull'ordine di una grandezza. Tuttavia, tali tempistiche restano ancora ben al di sotto di una soglia di percepibilità da parte di un ipotetico utilizzatore (si parla comunque di molto meno di un secondo di runtime).

Conclusioni

Questo lavoro ha mostrato la bontà e l'efficacia nell'utilizzo di tecniche di Machine Learning per la classificazione binaria di immagini biomedicali basata sull'uso di descrittori locali. Ognuno dei risultati ottenuti è largamente positivo, sia in termini di accuratezza che di generalità, nonché di tempi di esecuzione estremamente brevi.

Come ogni cosa a questo mondo, non è un modello perfetto, quanto sicuramente perfettibile.
Un passo ulteriore, infatti, per migliorare ciò che è stato esposto in questo elaborato, potrebbe essere l'utilizzo di tecniche automatiche, invece del tuning manuale, per la ricerca di parametri ottimali per ogni algoritmo. Un esempio è l'utilizzo della Grid Search, che valuta, a partire da una lista di valori specificati per i diversi parametri, la loro combinazione ottimale.
Un altro aspetto che si potrebbe valutare è l'uso di tecniche di analisi più avanzata, come le Curve di Apprendimento e le Curve di Convalida, che possono permetterci di capire meglio la tendenza di un algoritmo nel soffrire fenomeni di overfitting (o di underfitting), e quindi migliorare ulteriormente le prestazioni del modello.

In generale però c'è comunque da mettere in luce che si pone, già in questo stadio esposto, comunque come un grande passo in avanti rispetto alle tecniche classiche del settore medico per la diagnostica e la predizione di tumori al seno. Si può facilmente pensare alla

realizzazione di applicativi mobile/desktop basati su questa metodologia che possano essere di supporto all'attività del personale medico. Un esempio, infatti, potrebbe essere la realizzazione di un'interfaccia intuitiva e funzionale che consenta di inserire in ingresso un'immagine FNA che poi, una volta elaborata in cloud, restituirà la predizione accompagnata da statistiche di accuratezza e probabilità di FN e FP.

Bibliografia

[1] I numeri del cancro in Italia 2020,

https://www.aiom.it/wp-content/uploads/2020/10/2020_Numeri_Cancro-operatori_web.pdf

[2] Breast Cancer Wisconsin (Diagnostic) Data Set,

https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29

[3] S.W.Fletcher, W.Black, R.Harris, B.K.Rimer, and S.Shapiro. "*Report of the international workshop on screening of breast cancer*". Journal of the National Cancer Institute, 85:1644-1656, 1993.

[4] R.W.M.Giard and J.Hermans. "*The value of aspiration cytologic examination of the breasts. A statistical review of the medical literature*." Cancer, 69:2104-2110, 1992.

[5] Russell, Stuart; Norvig, Peter (2003) [1995]. Artificial Intelligence: A Modern Approach (4th ed.). Prentice Hall.

[6] Wikipedia, https://en.wikipedia.org/wiki/Machine_learning , 06-05-2021

[7] W.N. Street, W.H. Wolberg and O.L. Mangasarian. "*Nuclear feature extraction for breast tumor diagnosis*." IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.

[8] Scikit-learn, https://scikit-learn.org , 06-05-2021

[9] Sebastian Raschkase, "*Naive Bayes and Text Classification I, Introduction and Theory*", October 4, 2014, https://arxiv.org/pdf/1410.5329v3.pdf

[10] Agarap, Abien Fred. "*On breast cancer detection: an application of machine learning algorithms on the wisconsin diagnostic dataset*." ArXiv abs/1711.07831 (2018).

[11] Kohavi, Ron, "*A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*," da "International Joint Conference on Artificial Intelligence (IJCAI)", 14 (12): pp. 1137-43, 1995

[12] Google Colab, https://colab.research.google.com

[13] Tutte le figure presenti in questo elaborato sono state realizzate tramite il tool online Draw.io, disponibile al seguente link: www.draw.io