Analisi dei dati

Progetto

Luca Callisti

Università di Pisa
2023

Indice

0.1 Google Colab

Il link al Google Colab per il progetto sulla cirrosi è:
https://colab.research.google.com/drive/1yjiLNGMb2ydBiqYGUFSLhR0ZERi4kfmk?usp=sharing
Mentre il link per il progetto sul Covid è:
https://colab.research.google.com/drive/1v53nxtH5r3-Hu778sR7nBz1OFXae_WOO?usp=sharing

Capitolo 1

Cirrhosis Prediction Dataset

1.1 Introduzione

In questo proggeto alcuni predittori saranno usati su *Cirrhosis Prediction Dataset*[1] per capire in quale fase della malattia sono i pazienti.
Il dataset scelto contiene 424 dati di pazienti affetti da cirrosi bilineare primitiva che hanno partecipato ad uno studio randomizzato controllato con placebo del farmaco D-penicillamina. Della maggior parte dei pazienti abbiamo a disposizione i dati quasi completi mentre per 112 pazienti i dati raccolti sono solo parziali. Questo suggerisce nella fase di preprocessing dei dati di prestare particolare attenzione ai dati mancanti.

1.2 Analisi del Dataset

Nel dataset sono presenti 20 features di cui *Stage* sarà il nostro label mentre *ID* essendo una feature che non servirà nella predizione verrà tolta dal dataset.

ID	N_Days	Status	...	Prothrombin	Stage
1	400	D	...	12.2	4.0
2	4500	C	...	10.6	3.0
3	1012	D	...	12	4.0
4	1925	D	...	10.3	4.0
5	1504	CL	...	10.9	3.0

Figura 1.1

[1] fedesoriano. (August 2021). Cirrhosis Prediction Dataset. Retrieved [Date Retrieved] from https://www.kaggle.com/fedesoriano/cirrhosis-prediction-dataset.

Il dataset non contiene duplicati ed è composto da features sia numeriche che non numeriche. Inoltre ci deve aspettare delle entrate mancati ed infatti la figura seguente ci mostra il numero di entrate mancanti per ogni features.

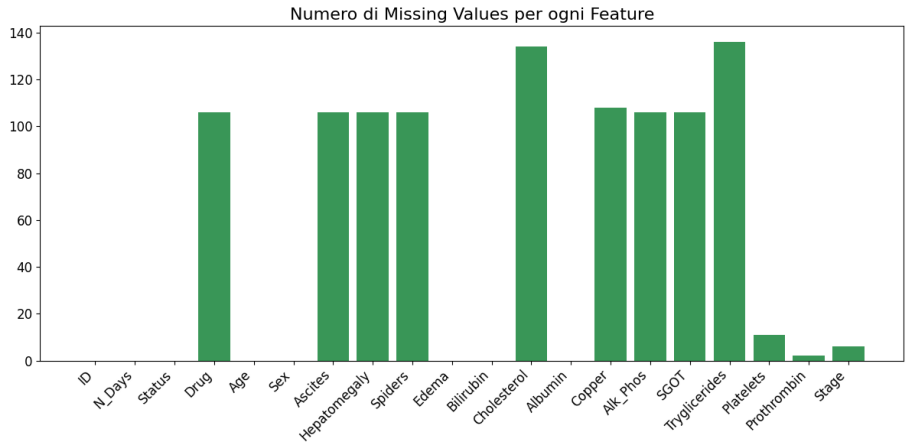


Figura 1.2

Come si nota dalla figura 1.2 in alcune categorie ho circa 100 dati mancanti che sono i 112 pazienti di cui abbiamo a disposizione solo i dati parziali. Per risolvere questo problema si potrebbe rimuoverli dal dataset ma poichè sono quasi un quarto ho preferito imputare i dati mancanti.
Dalla precendente figura si osserva che pure a *Stage* mancano 6 dati. Per capire come sostituirli vediamo come è distribuito il dataset rispetto alla categoria *Stage*.

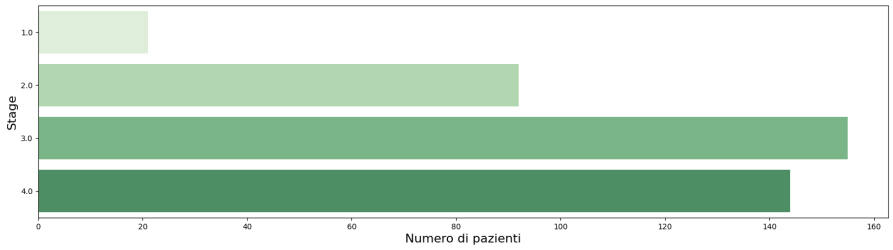


Figura 1.3

Si osserva che il dataset è fortemente sbilanciato per ciò si preferisce non assegnare nessun label ai dati mancanti ma rimuoverli dal dataset per non aggiungere bias. Essendo solo 6 ciò non influenzerà il risultato dello studio.
Analogamente anche per le categorie numeriche che hanno dei dati mancanti guardo come sono distribuiti.

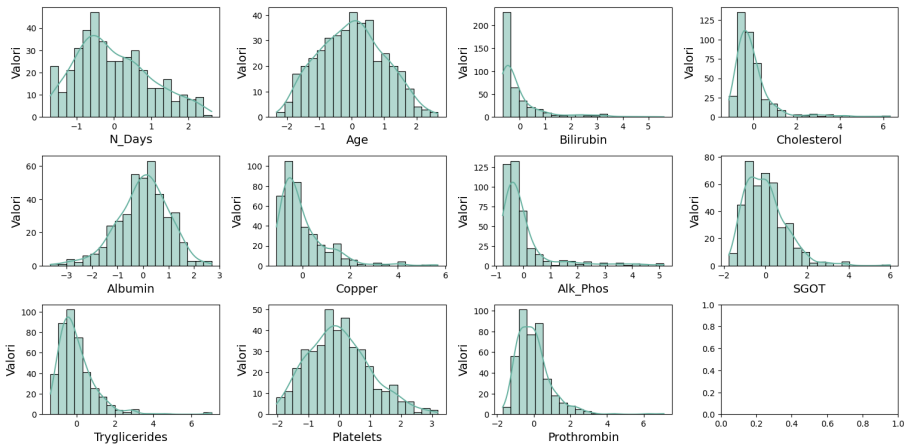
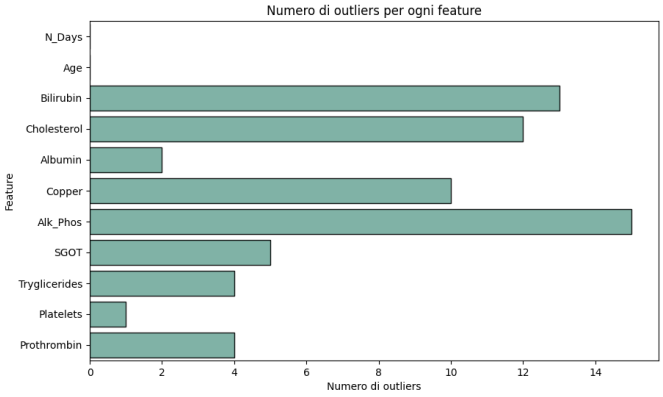


Figura 1.4

Dai grafici si può intuire la presenza di alcune anomalie, vorremmo quantificare il loro numero sia perchè possono influire sul risultato sia per decidere se andare a sostituire i valori mancanti con la media oppure con la mediana. Usando lo z-score e considerando outliers, quei valori che sono in valore assoluto più grande di 3, ho ottenuto il seguente grafico.



Mentre se si vuole fare uno studio usando i quartili si ottiene un risultato leggermente diverso ma che in ogni caso rivela la presenza di molti outliers, come si vede nella figura 1.5.

Nonostante la presenza di molti outliers nel dataset, non li ho rimossi perchè il loro numero sul totale non è indifferente e perchè, essendo dati medici, valori nelle analisi fortemente diversi potrebbero essere indicatore di un differente stadio della malattia.

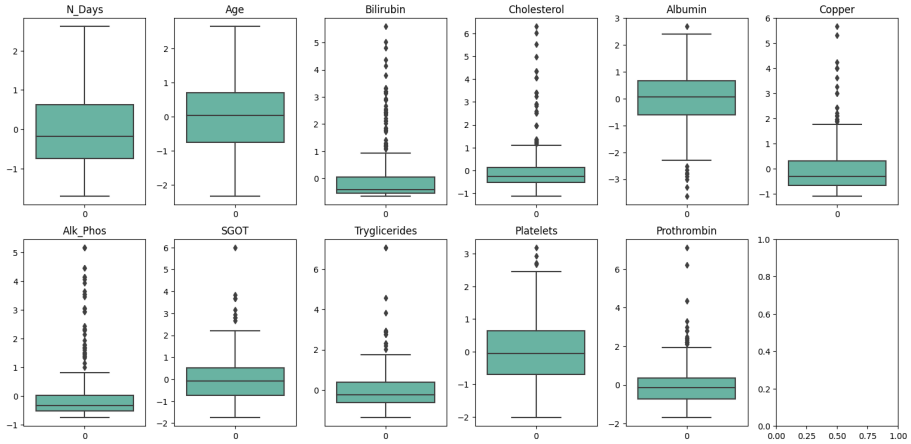
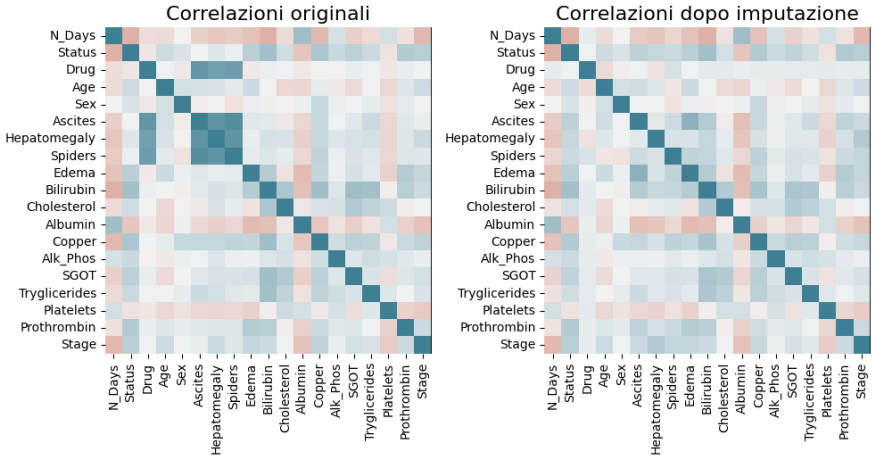


Figura 1.5

Lo studio dei dati anomali porta a sostituire i dati mancanti di tipo numerico con la mediana invece che la media perchè la mediana risente meno della presenza degli outliers e quindi sembra essere più adatta al nostro caso. Mentre per i dati di tipo non numerico useremo la moda.
Per verificare che la relazione tra le varie features non sia cambiata prima e dopo l'imputazione dei dati osserviamo la matrice di covarianza.



Si può osservare che la correlazione tra i dati è cambiata. Per evitare ciò ho provato ad eseguire una nuova imputazione dei dati sostituendo ai dati mancanti dei dati generati casualmente seguendo la distribuzione dei dati di cui conosco il valore. Ma anche in questo caso la matrice di correlazione non è simile a quella originale, forse ciò è dovuto al fatto che in alcune classi dovendo cambiare un quarto dei valori è probabile che cambino le relazioni.

1.3 Metriche

Nei problemi di classificazione grande attenzione si deve porre nella scelta delle metriche per calcolare le prestazione dei modelli, poichè sono utili per confrontare diversi classificatori, per trovare gli iperparametri migliori e per capire le carenze o i punti di forza di ogni modello.
In questo progetto, anche se non è una metrica, verrà usata la confusion matrix, che è molto utile per capire come il modello classifica i dati. Poi oltre all'accuratezza, che rappresenta la probabilità che il modello faccia una previsione corretta, saranno tenute in cosiderazioni anche la precisione, sensitività e specificità.
La prima ci indica quanto possiamo fidarci del modello quando predice un caso come positivo. La sensitività rappresenta la capacità del modello di individuare tutti i veri positivi. Infine la specificità rappresenta la capacità del modello di evitare falsi negativi.
A queste, che sono le principali metriche, ne possiamo aggiungere altre come l'accuratezza bilanciata che è la media delle sensitività calcolate per ciascuna classe, che rappresenta la probabilità con la quale un individuo di una classe può essere classificato correttamente. Questa metrica è particolarmente utile quando ci sono classi sbilanciate, dando più peso alle classi più piccole. Mentre se usata con classi abbastanza bilanciate l'accuratezza e l'accuratezza bilanciata tenderanno a convergere allo stesso valore.
Infine F1-score è la media armonica tra precisione e sensitività. Questa metrica premia modelli che hanno valori di precisione e sensitività simili e al contrario quando anche solo uno dei due è basso, F1-score sarà basso.
Poi per valutare le prestazioni di modelli per classificazioni multiclasse, si deve combinare i risultati ottenuti per ogni classe. Per fare ciò si usano diverse tipologie di medie come la micro average, in cui ogni esempio da lo stesso contributo alla metrica finale. La macro average, in cui ogni classe da lo stesso contributo alla metrica finale e la weighted average in cui ogni classe da un contributo secondo un peso per la metrica finale.
L'utilizzo di diverse metriche consente di ottenere una visione più completa del modello. In particolare con classi sbilanciate è consigliabile l'uso della macro average o weighted average per non far influenzare troppo il risultato dalle classi più grandi.
In particolare è importante osservare come micro f1, micro sensitivity, micro precision si riducano ad essere l'accuracy.

1.4 Predittori

1.4.1 Metodologia

Verranno usati come predittori Random Forest, Support Vector Machine e Neural Network e per allenarli si dividerà il dataset in un train set che sarà il 0.9 e il restante 0.1 sarà usato come test set. Siccome il dataset è sbilanciato, con il train set useremo la stratified 5-fold sia per valutare gli iperparametri sia per confrontare tra di loro i vari metodi e infine sarà usato il test set.
Come errori saranno calcolati l'accuratezza, l'accuratezza bilanciata, precisione e specificità, visto che in medicina è bene evitare che ci siano dei falsi negativi e falsi positivi, e F1-score. Siccome il dataset è molto sbilanciato la balanced accuracy sarà usata come parametro per scegliere gli iperparametri migliori per ogni modello.

1.4.2 Fase di allenamento

Per il Random Forest, per motivi computazionali ho fatto una ricerca degli iperparametri migliori fissando prima il numero degli alberi a 100 e cercando la miglior profondità e successivamente usata per cercare il miglior numero di alberi.

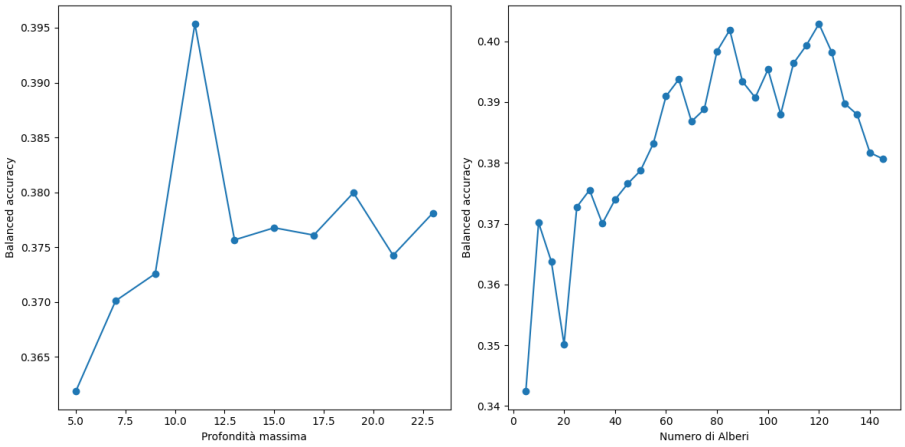


Figura 1.6

Alla fine il risultato migliore è stato dato da una profondità di 11 con 120 alberi, il fatto che gli alberi siano poco profondi è buono per evitare overfitting. Mentre per SVM abbiamo fatto una ricerca degli iperparametri facendo variare il kernel tra *poly, rbf* e *sigmoid* (per il kernel *poly* è stato impostato il grado uguale a 3) e il parametro c tra 0.1, 1, 10, 100, 1000.

Il migliore si è rivelato essere quello con kernel *sigmoid* e c=1000.
Mentre per il Neural Network per evitare l'overfitting sono state usate archi-

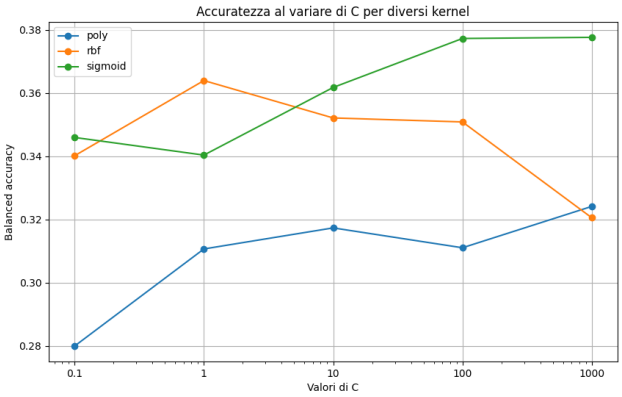


Figura 1.7

tetture con pochi neuroni e anche un Dropoutlayer con una probabilità del 0.2, inoltre le epoche sono state impostate a 50. In particolare ho usato le seguenti architetture $[32, 16]$, $[8, 8]$, $[16, 16]$, $[16, 8]$, con le funzioni di attivazioni *sigmoid*, *relu* e *tanh* per tutti gli strati tranne l'ultimo per cui è stata usata *softmax*.

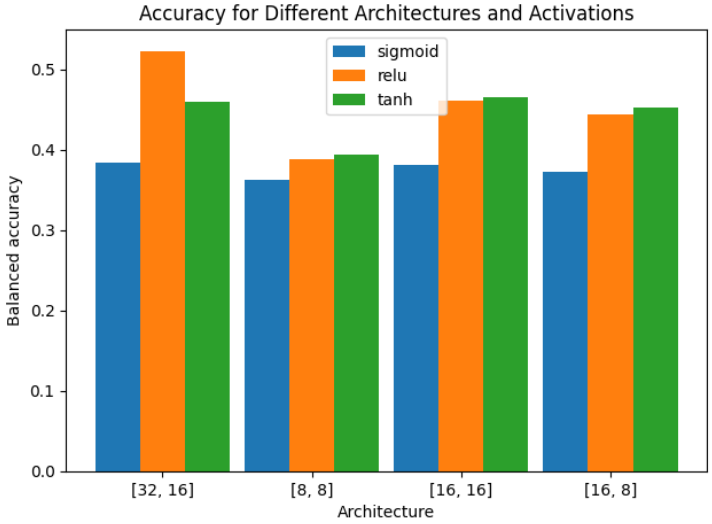


Figura 1.8

La migliore è stata l'archietettura con 32 neuroni nel primo layer e 16 nel secondo con l'activation function *relu*.

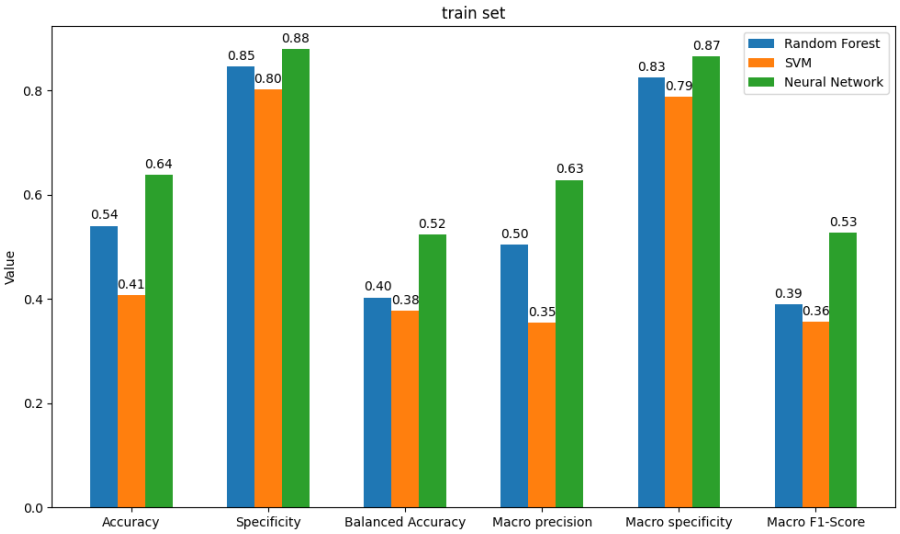


Figura 1.9

Confrontando le metriche dei predittori con gli iperparametri migliori, vediamo che la rete neurale é la migliore in ogni parametro, a seguire la random forest e infine la SVM.
In generale si osserva che sul train set tutti modelli hanno avuto una specificità particolarmente alta. Inoltre dal fatto che l'accuracy è più alta della balanced accuracy, si deduce che non tutte le classi sono state classificate bene nella stessa maniera. In particolare le classi più grandi hanno ottenuto risultati migliori.

1.4.3 Fase di test

Passiamo ora a vedere i risultati sul test set. Come previsto la prima classe ha

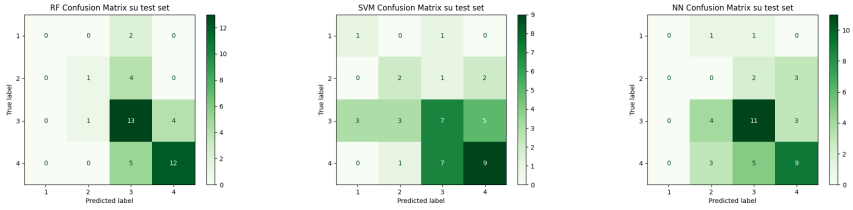


Figura 1.10

avuto difficoltà ad essere classificata da ogni predittore, solo la SVM è riuscito a classificarne bene uno, mentre le classi con i migliori risultati sono le ultime

due.

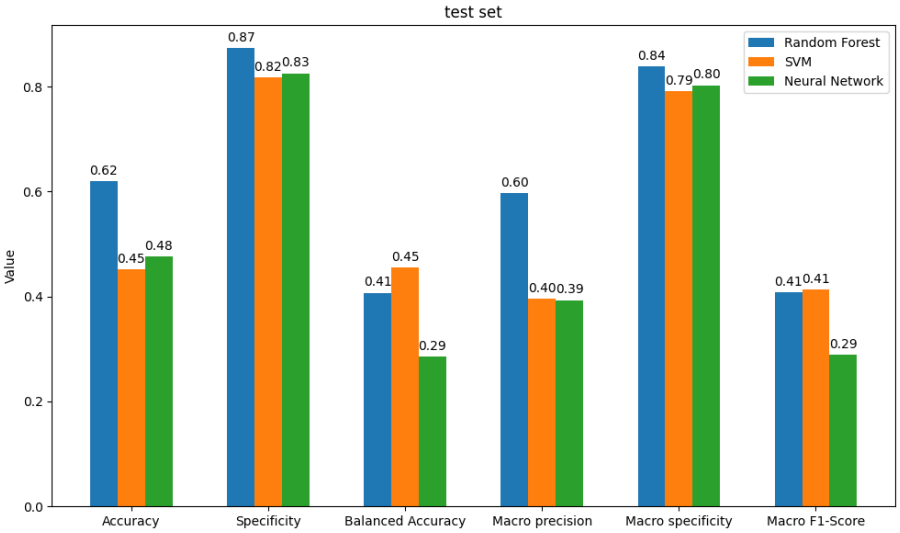


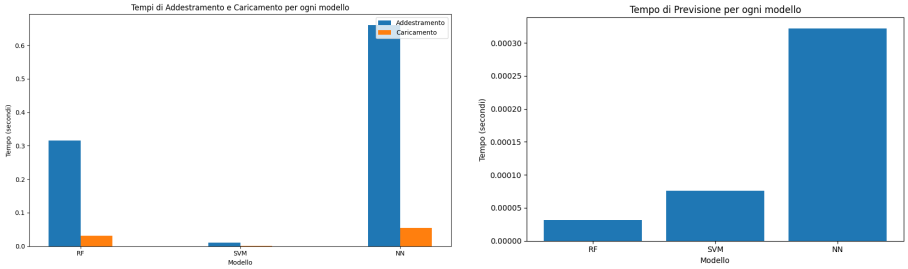
Figura 1.11

Confrontando le figure 1.9 e 1.11 si può osservare come la rete neurale sia andata in overfitting, infatti i parametri della accuracy e balanced accuracy sono calati entrambi di molto.
La SVM ha avuto un leggero miglioramento nell'accuracy e, nonostante abbia il valore più basso in questa categoria, mentre è quella con il valore più alto nella balanced accuracy. Ciò è dovuto al fatto che è stata l'unica a classificare bene un elemento della prima classe.
Mentre la Random Forest è il modello che ha avuto il miglioramento maggiore, ma dalla confusion matrix si capisce che classifica bene solo le ultime due classi. Ciò ci suggerisce che se nel test set le proporzioni delle classi fossero diverse si otterrebbe l'accuracy sarebbe più bassa.
Infine, come anche nella fase di train, tutti i modelli hanno una specificità altà quindi sono in grado di evitare falsi negativi.

1.4.4 Complessità

Per quanto riguarda la complessità computazionale ho preso in considerazioni tre parametri. Il tempo che ogni modello impiega ad addestrarsi con i parametri migliori, il tempo che impiega a caricarsi e il tempo che impiega per fare le previsioni.

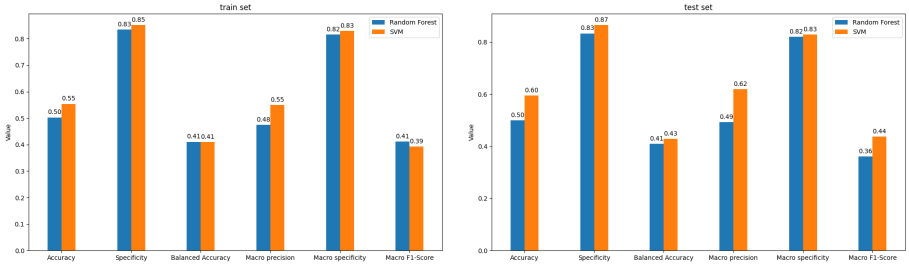
Si osserva che la Rete Neurale è molto più lenta rispetto alle altre, al contrario SVM è il modello che si addestra e viene caricato più velocemente anche se per il tempo di previsione non è il più veloce, infatti è superato dalla random forest.



Anche se questi dati devono essere presi con cautela visto che per ottenere dei risultati apprezzabili per la previsione ho calcolato il tempo che ogni modello impiegava a dare una etichettare ogni elemento del dataset e poi diviso per la grandezza del dataset stesso.
In effetti se indichiamo con n la grandezza del train set con m il numero di features allora si ha che il tempo per allenare SVM è $\mathcal{O}(n^2)$ per il random forest è $\mathcal{O}(n * \log(n) * m * k$ con k =numero di decision tree.

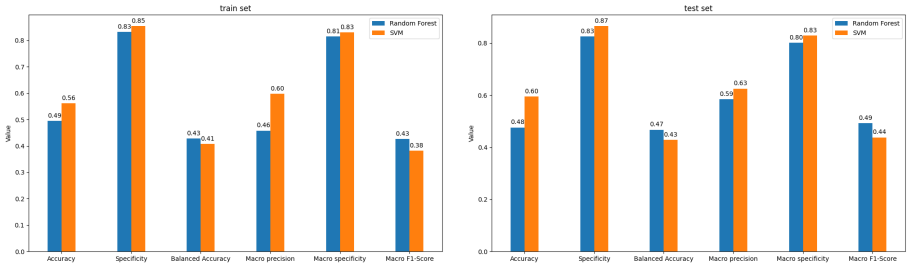
1.4.5 Features selection

Un'altra fase di selezione dei modelli è la features selection nella quale si valuta quali categorie è meglio considerare per migliorare il nostro modello. Siccome per trovare la miglior combinazione dovremmo fare troppi tentativi e non sarebbe computazionalmente fattibile. Farò la forward selection e backward selection che richiedono meno iterazioni e solo sul Random Forest e SVM, sempre per motivi di complessità.
Partendo con la backward selection si ottiene che entrambi i metodi scelgono *N_Days, Hepatomegaly, Albumin, Alk_Phos, SGOT, Trygliceride*s, *Prothrombin* a queste 7 la Random forest aggiunge *Ascites, Spiders, Cooper* mentre la SVM sceglie *Drug, Edema, Birilium, Cholesterol, Platelets* per avere un totale di 10 e 12 features.



Mentre nella forward selection le features in comune sono *Prothrombin, Albumin, Hepatomegaly, 'N_Days', 'Alk_Phos', Status* oltre a queste la random forest sceglie anche *SGOT, Spiders, Cholesterol, Sex, Ascites, Drug* per un to-

tale di 12 features, mentre la SVM sceglie *Copper, Platelets* per arrivare a 8 features.



Osserviamo che *N_days, Hepatomegaly, Albumin, Prothrombin* che sono state sempre selezionate erano quelle che nella matrice di correlazioni presentavano i valori più in valore assoluto.
Mentre per le prestazioni dei modelli si hanno dei comportamenti diversi. Infatti nonostante che la balanced accuracy nel train set aumenta sia per la Random forest che per la SVM, poi nel test set la random forest ottiene dei risultati peggiori. Mentre la SVM sembra guadagnare accuratezza quando la dimensione dei dati diminuisce.

1.5 Conclusione

Ho usato diverse metriche di accuratezza per studiare meglio i modelli e avere una visione chiara delle loro prestazioni.
In generale per problemi multiclasse sbilanciati è importante usare la macro avarage o la balanced accuracy per dare la stessa importanza a tutte le classi, inoltre anche l'utilizzo della confusion matrix aiuta ad avere un quadro generale migliore.
Guardare solo l'accuratezza potrebbe fornire una visione parziale delle prestazioni del metodo in questione. Ad esempio guardare la specificità è importante se ci sono differenze significative dei costi degli errori, come nell'ambito diagnostico.
Nonostante i pochi dati, i modelli presentano tutti una specificità alta e in ambito medico avere una specificità alta è importante per evitare di classificare soggetti sani come malati. I predittori hanno una accuratezza non troppo elevata, anche se con i pochi dati a disposizione non è da considerare un cattivo risultato.
Dalla analisi della complessità si ha che la rete neurale è il metodo più lento e che ottiene dei risultati scarsi. Mentre la random forest è il modello che si comporta meglio sul test set, ma il suo tempo di addestramento è non di poco superiore a quello della SVM.
Quindi se si vuole tenere basso il costo computazionale una buona ideaa può essere quella di usare la SVM con la features selection, che come visto sopra, gli fa ottenere delle prestazioni migliori.

Capitolo 2

Covid-19 Image Dataset

2.1 Introduzione

Passiamo ora ad analizzare un secondo dataset che contiene immagini di radiografie del torace di alcuni pazienti.
Il dataset in questione è *Covid-19 Image Dataset*[1] e l'obiettivo è distinguere i pazienti malati di covid, quelli che hanno una polmonite virale e quelli sani. Sono presenti in tutto 317 radiografie già divise in train e test set.

2.2 Analisi del Dataset

Come già detto il dataset è già diviso in test set e train set e dalla seguente tabella 2.1 possiamo vedere come sia leggermente sbilanciato, poichè la classe Covid ha qualche esempio in più. Ciò non influirà nello studio.

Classe	Dimensione	
	Train set	Test set
Normal	70	20
Viral Pneumonia	70	20
Covid	111	26

Tabella 2.1: Dimensioni dei test set e train set per le classi

Guardando le immagini si può notare che hanno dimensioni diverse, anche se sono tutte tendenzialmente quadrate. Per questo motivo per operare con dimensioni standard ho ridimensionato le immagini, creando due nuovi dataset. Il primo dataset contiene immagini con 512×512 pixels mentre il secondo 256×256 pixels.

[1] Retrieved from https://www.kaggle.com/datasets/pranavraikokte/covid19-image-dataset.

Per avere un'idea delle risoluzioni delle nuove immagini si può vedere la figura 2.1, nella quale si può notare che c'è una perdita di qualità in particolare in quella a Dx. Nonostante ciò per motivi computazionali la riduzione delle dimensioni delle immagini era necessaria.

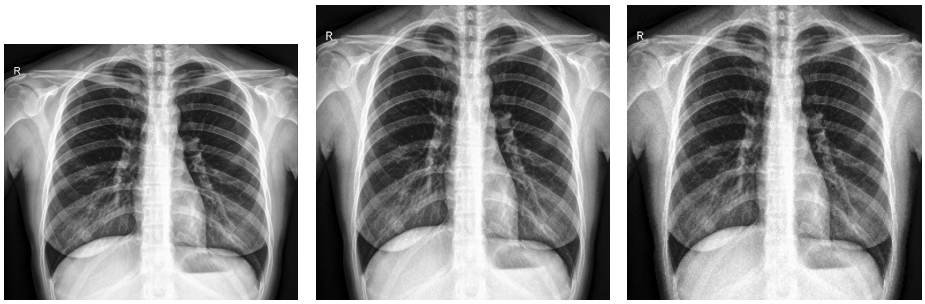


Figura 2.1: A sinistra è presente l'immagine originale, nel centro quella con 512×512 px e a destra quella con 256×256 px

Infine le radiografie dei pazienti sani sono state etichettate con 0, quelle dei pazienti affetti da polmonite con 1 e le radiografie delle persone con il covid con 2. In seguito userò questa notazione.

2.2.1 Metodologia

Per analizzare i dati sono stati usati due predittori: Support Vector Machine e Neural Network.
Uno dei principali vantaggi del SVM è che riesce a gestire dati con dimensione alta e tende ad andare meno in overfitting rispetto alle reti neurali. Per questo mi sembra adatto per classificare le immagini, anche perchè richiedendo risorse computazionali non così alte ho potuto usare il dataset con la risoluzione più alta.
Mentre per la rete neurale ho usato una convolutional neural network, in cui i layer convoluzionali sono stati alternati con layer di maxpooling, per ridurre il numero di parametri. Infine è stato usato un dropout per ridurre l'overfitting e come ultimo strato è stato usato un layer completamente connesso. Le funzioni di attivazioni sono state usate come iperparametro tranne quella dell'ultimo layer che è *softmax*, mentre per l'addestramento ho usato 10 epoche.
Inoltre data la complessità computazionale per confrontare gli iperparametri e i metodi ho usano *hold-out-set*, usando il 20% dei dati del train set come validation set.
Per confrontare i metodi è stata usata la micro average e la weighted average prendendo come pesi il reciproco della frequenza di ogni label ed è stata usata l'accuracy per confrontare i vari iperparametri.

2.2.2 Fase di allenamento

Per SVM ho fatto una ricerca degli iperparametri facendo variare il kernel tra *poly, rbf* e *sigmoid* (per il kernel *poly* è stato impostato il grado uguale a 3) e il parametro c tra 0.1, 1, 10, 100, 1000.

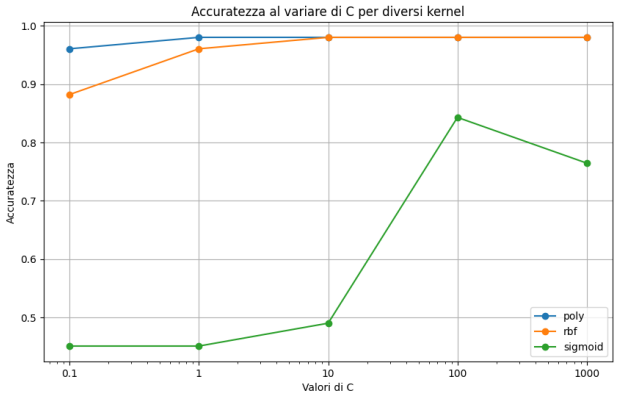


Figura 2.2

Nella figura 2.3 si vede come la micro accuracy al variare dei parametri assume il risultato migliore con *poly* e *c=1*.
Mentre per la rete neurale per gli strati convoluzionali sono stati usate le seguenti architetture $[32, 32, 64]$, $[32, 64, 128]$ e $[64, 64, 128]$ mentre come funzioni di attivazioni sono state usate *sigmoid, relu* e *tanh*.

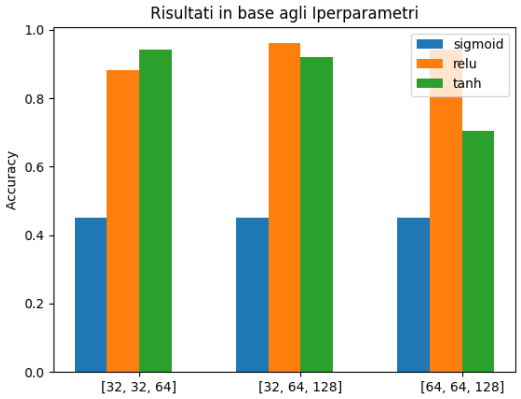


Figura 2.3

La configurazione migliore è stata [32, 64, 128] con la funzione *relu*.
Confrontando le configurazioni migliori e si può vedere come entrambe abbiano ottenuto degli ottimi risultati in tutte le categorie. In particolare la SVM assume sempre il massimo, mentre la rete neurale è subito sotto.

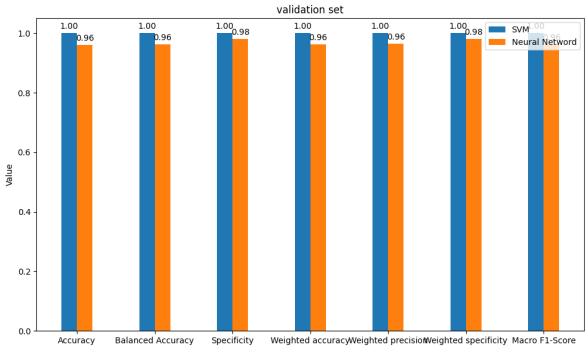


Figura 2.4

2.2.3 Fase di test

Per la fase di test entrambi i metodi sono stati allenati con i parametri migliori sull'intero train set e poi valutati sul test set.
Possiamo vedere che gli alti valori ottenuti nella fase di addestramento sono leggermente calati. Per la SVM di circa 0.09 mentre la rete neurale di circa 0.13. Ciò significa c'è stato overfitting.
Dalla confusion matrix si osserva che la SVM etichetta bene tutte le classi mentre la rete neurale ha qualche problema in più nella classe 1.

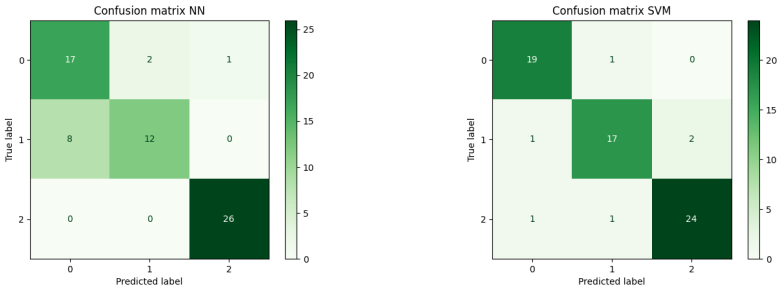


Figura 2.5

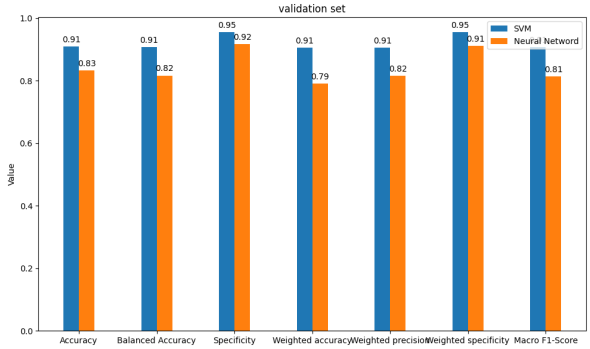


Figura 2.6

2.2.4 Complessità

Analogamente a prima per la complessità computazionali ho preso in considerazione tre parametri: il tempo di addestramento, il tempo che impiega a caricarsi e il tempo impiegato a fare previsioni.

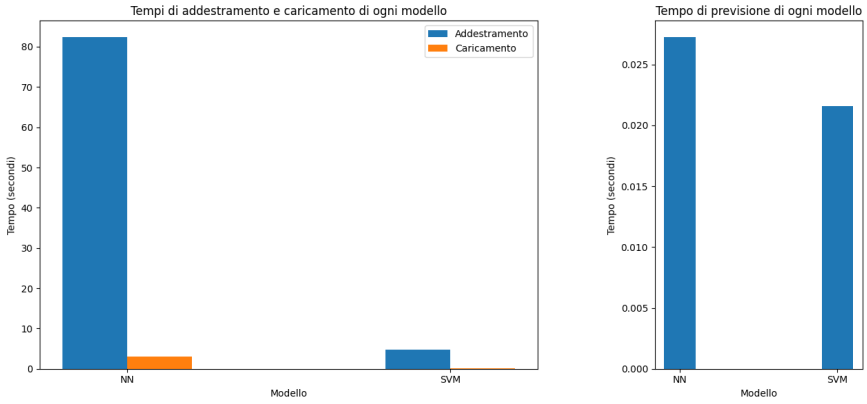


Figura 2.7

Come ci si poteva aspettare la rete neurale impiega molto più tempo ad addestrarsi, mentre il tempo per fare previsioni è sostanzialmente uguale.

2.3 Conclusioni

Entrambi i metodi risultano ottimi per affrontare questo tipo di problema. La SVM presenta delle valutazioni leggermente più alte e ciò è probabilmente dovuto al fatto che sta analizzando le immagini con più dettagli.

Suppongo che una rete neurale allenata su delle immagini con una risoluzione più alta otterrebbe risultati migliori.
Sicuramente uno dei dati più sorprendenti sono le prestazioni ottenute dalla SVM se consideriamo il tempo di allenamento che è molto più basso rispetto alla rete neurale.

Bibliografia

[1] Margherita Grandini, Enrico Bagli e Giorgio Visani. "Metrics for multi-class classification: an overview". In: *arXiv preprint arXiv:2008.05756* (2020).

[2] Shai Shalev-Shwartz e Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Publisher Name, Year. URL: `https://www.example.com/book-url`.

[3] *Time Complexity of ML Models*. URL: `https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770`.