Università degli Studi di Milano

Facoltà di Scienze e Tecnologie

Corso di Laurea in Informatica

MODELLI DI PREVISIONE DEL RISCHIO DI MORTALITÀ PER PAZIENTI COVID

Relatore: Prof.ssa Elena Casiraghi

Correlatore: Prof. Dario Malchiodi

Correlatore: Postdoc. Alessandro Petrini

Tesi di: Marco CAVALLARI

Matricola: 923537

Anno Accademico 2020-2021

Alla mia famiglia.

1 Sommario

Il lavoro da me svolto durante il periodo tirocinio riguarda l'ambito dell'apprendimento automatico. Lo scopo è stato quello di sviluppare un metodo supervisionato per la predizione del rischio di mortalità per pazienti COVID-19, al fine di individuare per tempo pazienti positivi alla malattia e soprattutto capire quanto la malattia fosse grave per i singoli pazienti. Per lo sviluppo del progetto è stato utilizzato un dataset proveniente da due ospedali cinesi: l'Union Hospital (che ha fornito i dati di 1126 pazienti) e il Liyuan Hospitali (che ha fornito i dati di altri 295 pazienti). Il lavoro è iniziato con l'imputazione dei valori mancanti del dataset tramite l'algoritmo 'MissForest'. Un valore mancante può avere diversi significati. È possibile che il campo non fosse applicabile, che l'evento non si sia verificato o che i dati non fossero disponibili. Potrebbe essere accaduto che la persona che ha immesso i dati non conoscesse il valore corretto o non abbia verificato l'effettiva compilazione di un campo[12] . Missforest è un algoritmo di imputazione basato sull'utilizzo di Random Forest. Inizialmente i valori mancanti del dataset sono stati sostituiti con la media delle osservazioni di quella variabile (per le variabili continue) o con la sua classe più frequente (per le variabili categoriche). Dopo di che, per ogni variabile (che all'interno dell'elaborato ho anche chiamato coi termini feature, attributo e caratteristica) avente valori mancanti, MissForest addestra una random forest sulla parte di dati non mancanti e prevede i mancanti della medesima variabile. Quando non sono più presenti dati mancanti per tutte le variabili, viene completata un'iterazione. Le imputazioni continuano per più iterazioni (vedremo più avanti il perché).

Una volta ottenuto un dataset privo di valori mancanti, sono stati applicati e confrontati due algoritmi di feature selection: Boruta e MRMR. La "Feature selection" è il processo di riduzione del numero di variabili di input durante lo sviluppo di un modello predittivo. Gli algoritmi di feature selection possono essere divisi in "minimal-optimal" o "all-relevant". Boruta è un algoritmo del tipo "all-relevant", ovvero trova un sottoinsieme di caratteristiche dal dataset che, individualmente, sono rilevanti per una data classificazione. MRMR (chè è un algoritmo minimal optimal) invece cerca di identificare un insieme di caratteristiche che nella loro totalità e non nella singolarità abbiano il massimo potere predittivo possibile. Con Boruta, le feature non competono tra loro ma con una versione randomizzata di loro stesse (le shadow feature; anche questo argomento verrà trattato a dovere nei capitoli successivi). Quando l'importanza di una feature del dataset è superiore a quella della shadow feature più importante, abbiamo una "hit", ovvero la feature in questione viene scelta da Boruta. Con MRMR invece, viene utilizzata la regola di selezione "Maximum Relevance - Minimum Redundancy" ovvero si seleziona ad ogni iterazione la feature che ha la massima rilevanza rispetto alla variabile target e la minima ridondanza rispetto alle caratteristiche che sono state già selezionate nelle iterazioni precedenti. Oltre all'analisi di questi due algoritmi, sono stati anche analizzati i casi in cui venivano prese le feature selezionate dall'unione e intersezione dei due algoritmi sopra citati.

Finita la fase di feature selection, è stato implementato un autoencoder, un tipo di rete neurale composta da due sub-model: un encoder e un decoder. Una volta che l'autoencoder è stato addestrato, la parte di decoder è stata scartata ed è stato utilizzato solo il modello fino al bottleneck (l'encoder). L'output dell'encoder è un vettore a lunghezza fissa che fornisce una rappresentazione compressa dei dati di input. L'encoder addestrato è stato poi utilizzato per comprimere i dati di input e addestrare un modello predittivo, in questo caso una semplice rete neurale. L'obiettivo è stato quello di addestrare una rete neurale con i dati compressi e una rete

neurale con i dati originali del dataset e vedere se si otteneva una sorta di miglioramento con i dati compressi provenienti dall'encoder.

Le tecniche di feature selection e addestramento della rete neurale citata nel precedente capoverso sono state sperimentate applicando k-fold cross-validation. I risultati dei modelli predittivi sviluppati sono stati valutati tramite metriche quali F1-score, accuratezza, precisione, recall e ROC AUC.

L'elaborato è organizzato in tre capitoli. Il primo capitolo contiene l'introduzione all'elaborato dove viene fatto un riassunto di come il COVID-19 abbia cambiato le nostre vite e di come esso abbia anche impattato il mondo scientifico, soprattutto quello medico e informatico. Il secondo capitolo è inerente al background teorico e spiega la teoria che sta alla base di ciò che è stato poi messo in pratica. L'ultimo capitolo è dedicato alla descrizione di come sono state implementati i metodi descritti nel capitolo precedente e di quali risultati si sono ottenuti, anche attraverso l'utilizzo di tabelle e figure.

Indice

2 Introduzione

Per via dell'emergenza COVID-19, i ricercatori nel campo dell'apprendimento automatico e della radiologia si sono affrettati a sviluppare algoritmi che potrebbero aiutare con diagnosi, triage e gestione della malattia. Di conseguenza, migliaia di modelli diagnostici e prognostici mediante radiografie del torace e CT e analisi dei dati clinici sono stati sviluppati. [1]

CT sta per "tomografia computerizzata" ed è una tecnica di indagine radiodiagnostica (diagnostica per immagini), con la quale è possibile riprodurre immagini in sezione (tomografia) e tridimensionali dell'anatomia, create da un'analisi generata al computer, dell'attenuazione di un fascio di raggi X mentre passa attraverso una sezione corporea.[14]

2.1 COVID-19: Cos'è

Il COVID-19 è un virus infettivo emerso in Cina a fine dicembre 2019 che si è diffuso in almeno 213 paesi in tutto il mondo. Pertanto, l'11 marzo 2020, l'OMS ha dichiarato l'epidemia di COVID-19 come pandemia.

Il COVID-19 è il nome della sindrome provocata dal coronavirus Sars (Severe acute respiratory syndrome) che è lo specifico coronavirus che sta causando l'epidemia.

Oltre a colpire gli esseri umani, questa enorme famiglia di virus chiamati coronavirus colpisce principalmente vari animali come pipistrelli, gatti e cammelli. Questi animali aventi una malattia della famiglia coronavirus possono infettare un essere umano che a sua volta può quindi diventare una causa di trasferimento orizzontale (da uomo a uomo).

La complessità clinica di COVID-19 varia da casi asintomatici a casi nella quale si manifesta polmonite grave la cui progressione verso l'insufficienza respiratoria è difficile da prevedere. La polmonite si verifica principalmente nella seconda o terza settimana di un'infezione sintomatica ed è caratterizzata da un tasso di mortalità del 3-10%, che aumenta il rischio di insufficienza multiorgano e ventilazione meccanica.

I pazienti riferiscono più comunemente l'insorgenza improvvisa di dispnea durante le attività quotidiane o il riposo. I segni clinici importanti includono frequenza respiratoria ≥ 30 respiri al minuto, saturazione di ossigeno nel sangue ≤ 93%, pressione parziale di ossigeno arterioso per frazione del rapporto di ossigeno inspirato (PaO2/FiO2) < 300 mmHg.

Recapitolando, questi sono i principali sintomi di pazienti COVID-19:

- Sintomi più comuni: febbre, tosse secca e spossatezza,
- Sintomi meno comuni: indolenzimento e dolori muscolari, mal di gola, diarrea, congiuntivite, mal di testa perdita del gusto o dell'olfatto e eruzione cutanea o scolorimento delle dita di piedi o mani,
- Sintomi gravi: difficoltà respiratoria o fiato corto, oppressione o dolore al petto e perdita della facoltà di parola o di movimento.

Nel complesso, vi è un alto grado di incertezza sia nella progressione dello stato di salute del paziente sia nella velocità con cui i pazienti sviluppano insufficienza respiratoria che richiede ventilazione meccanica. [2]

2.2 L'impatto della pandemia sul mondo scientifico

La pandemia di COVID-19 ha trovato il sistema sanitario inadeguatamente preparato in praticamente tutto il globo e ha sollecitato la necessità di nuovi strumenti per affrontare questa emergenza sanitaria e clinica senza precedenti. I metodi di machine learning hanno mostrato il potenziale per produrre modelli predittivi che possono essere applicati per assistere e migliorare le decisioni cliniche per un'ampia varietà di risultati e sono stati recentemente utilizzati in risposta all'emergenza COVID-19. [3]

Ad esempio, cinque modelli, tra cui random forest, XGBoost, percettrone, vector machines e logistic regression sono stati utilizzati e analizzati. Da questa analisi ne è risultato che Il modello XGBoost è il migliore nel trovare gli anticorpi per il virus COVID-19. [4]

I metodi di apprendimento automatico si sono dimostrati valide strategie anche per valutare dati ad alta dimensionalità come immagini mediche. I risultati della TC o dei raggi X dei pazienti COVID-19 hanno somiglianze con altre malattie della polmonite atipiche e virali. Pertanto, i metodi di apprendimento automatico potrebbero facilitare la discriminazione automatica di COVID-19 da altre condizioni di polmonite.

Diversi modelli, come Ensemble, VGG-16, ResNet, InceptionNetV3, MobileNet v2, Xception Net e Truncated Inception Net, sono stati utilizzati allo scopo di valutare le immagini del torace dei pazienti COVID-19. In particolare, l'applicazione di questi metodi aventi come input radiografie hanno offerto risultati promettenti. [13]

2.3 Scopo dell'elaborato

Lo scopo dell'elaborato è quello di studiare e applicare modelli di machine learning ad un dataset medico di provenienza cinese di cica 1500 pazienti. Nel dettaglio ciò che ho fatto è stato quello di:

1) "Pulire" il dataset,
2) Utilizzare l'algoritmo Miss Forest per imputare i valori mancanti del dataset,
3) Applicare diversi algoritmi di feature selection in modo da comprendere quali feature fossero realmente importanti per il nostro scopo, ovvero predire il rischio legato a COVID-19,
4) Sviluppare un autoencoder (dalla quale tratterò più in dettaglio nei seguenti capitoli) e utilizzarlo per comprimere i dati di input e allenare un modello di apprendimento automatico diverso,
5) Utilizzare una semplice rete neurale e addestrarla con e senza l'utilizzo di dati compressi e sui diversi dataset derivanti dalla feature selection fatta in precedenza.

3 Nozioni teoriche

3.1 Algoritmi di Machine Learning

L'apprendimento automatico (ML) è lo studio di algoritmi informatici che migliorano automaticamente attraverso l'esperienza e l'uso dei dati. I modelli effettuano delle previsioni sulla base dei dati che gli vengono passati come input, e questi stessi modelli sono generati tramite allenamento utilizzando algoritmi di apprendimento.

3.2 Alberi di decisione e Random Forest

Random Forest è un algoritmo di apprendimento automatico supervisionato in grado di affrontare sia compiti di classificazione che di regressione. Prima di parlare di classificazione e regressione è necessario introdurre il concetto di modellazione predittiva e di "Function approximation").

La modellazione predittiva è il problema che consiste nello sviluppare un modello utilizzando dati storici per fare una previsione su nuovi dati in cui non abbiamo la risposta. La modellazione predittiva può essere descritta come il problema matematico che consiste nell'approssimare una funzione di mappatura (f) dalle variabili di input (X) alle variabili di output (y). Questo è chiamato Function approximation. La modellazione predittiva di classificazione è il compito di approssimare una funzione di mappatura (f) dalle variabili di input (X) alle variabili di output discrete (y). Le variabili di output sono spesso chiamate etichette o categorie. La funzione di mappatura prevede la classe o la categoria per una data osservazione. La modellazione predittiva di regressione è il compito di approssimare una funzione di mappatura (f) dalle variabili di input (X) a una variabile di output continua (y). Una variabile di output continua è un valore reale, come un numero intero o un valore in virgola mobile. Si tratta spesso di quantità, come quantità e dimensioni.[15]

Una RF è composta da un insieme di alberi decisionali. Un albero decisionale è una struttura ad albero in cui ogni nodo interno denota un test su un attributo, ogni ramo rappresenta un risultato del test e ogni nodo foglia indica la classe di appartenenza del gruppo arrivato a tale nodo.

RF è un algoritmo di apprendimento automatico basato su alberi che sfrutta la potenza di più alberi decisionali per prendere decisioni. La *Figura 1* illustra quanto appena detto.

Ma perché la chiamiamo “random” forest? Questo perché gli alberi decisionali vengono creati casualmente. Ciascun nodo nell'albero decisionale lavora su un sottoinsieme casuale di feature per calcolare l'output. La foresta casuale combina quindi l'output dei singoli alberi decisionali per generare l'output finale. [5]

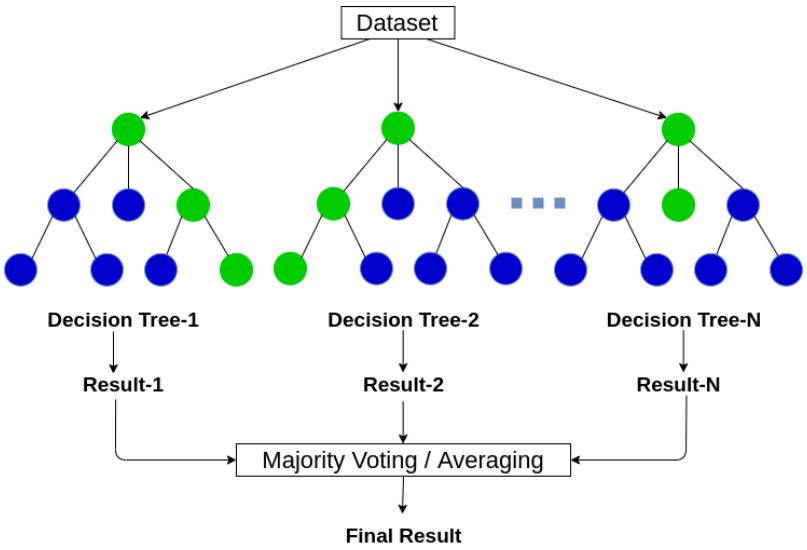


Figura 1: rappresentazione di Random Forest

3.3 Imputazione di valori mancanti tramite Miss Forest

'MissForest' è un algoritmo utilizzato per imputare valori mancanti, in particolare nel caso di dataset contenenti dati di diverso tipo. Può essere infatti utilizzato per imputare dati continui e/o categorici. Inizialmente questo algoritmo imputa tutti i dati mancanti utilizzando la media/moda (a seconda se la feature è di tipo continuo o categorica). Dopo di che, per ogni variabile avente valori mancanti, MissForest addestra una random forest sulla parte di dati non mancanti e prevede i mancanti della medesima variabile. Questo processo di addestramento e previsione si ripete in un processo iterativo fino a quando non viene soddisfatto un criterio di arresto o non viene raggiunto un numero massimo di iterazioni specificato dall'utente. Le *Tabelle1-2-3-4* mostrano un esempio di come Miss forest lavori.

X_1	X_2	X_3
21	Small	1.3
21	Large	1.5
#N/A	Medium	1.3
18.8	Medium	1.3
22.8	Medium	1.3
13.3	#N/A	1.6
15.8	#N/A	1.9

Tabella 1

X_1	X_2	X_3
21	Small	1.3
21	Large	1.5
18.78	Medium	1.3
18.8	Medium	1.3
22.8	Medium	1.3
13.3	Medium	1.6
15.8	Medium	1.9

Tabella 2

X_1	X_2	X_3
21	Small	1.3
21	Large	1.5
19.9830	Medium	1.3
18.8	Medium	1.3
22.8	Medium	1.3
13.3	Medium	1.6
15.8	Medium	1.9

Tabella 3

X_1	X_2	X_3
21	Small	1.3
21	Large	1.5
18.78	Medium	1.3
18.8	Medium	1.3
22.8	Medium	1.3
13.3	Small	1.6
15.8	Medium	1.9

Tabella 4

Le *Tabelle 1,2,3 e 4* funzionano nella seguente maniera:

- *Tabella **1*** mostra il dataset originale con dei valori mancanti,
- *Tabella **2*** mostra come i valori mancanti vengano sostituiti con media/moda. Inoltre una prima random forest viene costruita per la predizione di X_1,
- *Tabella **3*** mostra come i valori mancanti di X_1 siano stati sostituiti con quelli predetti dalla random forest. Inoltre una seconda random forest viene costruita per la predizione di X_2,
- *Tabella **4*** mostra come i valori mancanti di X_2 siano stati sostituiti con quelli predetti dalla random forest.

Il motivo per le iterazioni multiple è che, dall'iterazione 2 in poi, le random forest che eseguono l'imputazione verranno addestrate su dati di qualità sempre migliore che a loro volta sono stati imputati all'iterazione precedente. [6]

Vediamo quanto appena spiegato con un ulteriore esempio pratico (*Tabelle 5-10*):

Weight	Height
2.6	47.1
5.3	Nan
6.7	64.7
8.4	Nan
7.4	68.2
10.1	81.6

Tabella 5: dataset con valori mancanti

Weight	Height
2.6	47.1
5.3	65.4
6.7	64.7
8.4	65.4
7.4	68.2
10.1	81.6

Tabella 6: valori mancanti imputati con media

Weight	Height	
2.6	47.1	Training set
5.3	65.4	Prediction Set
6.7	64.7	Training set
8.4	65.4	Prediction set
7.4	68.2	Training set
10.1	81.6	Training set

Tabella 7: divisione train/test

Weight	Height	
2.6	47.1	
5.3	59.1	RF Prediction
6.7	64.7	
8.4	77.4	RF Prediction
7.4	68.2	
10.1	81.6	

Tabella 8: dati imputati all'iterazione 2(prima iterazione dopo che i valori mancanti sono stati imputati tramite media)

Weight	Height	
2.6	47.1	Training set
5.3	59.1	Trainng set
6.7	64.7	Training set
8.4	74.4	Prediction set
7.4	68.2	Training set
10.1	81.6	Training set

Tabella 9: l'algoritmo decide nell'iterazione corrente (3) di regolare le previsioni fatte o mantenerle. In questo caso la prima viene mantenuta e la seconda viene regolata

Weight	Height
2.6	47.1
5.3	59.1
6.7	64.7
8.4	73.9
7.4	68.2
10.1	81.6

Tabella 10: in questo caso l'esecuzione termina all'iterazione 4

Il processo di imputazione viene ripetuto fino a quando la somma delle differenze al quadrato tra i risultati dell'imputazione corrente e quelli precedenti non aumenta e MissForest restituisce l'imputazione precedente come risultato finale.

Per terminare, ecco lo pseudo-codice di miss forest (*Algoritmo 1*).

1. Make an **initial guess** for all missing categorical/numeric values (e.g. mean, mode)
2. $k \leftarrow$ vector of column indices in X, sorted in **ascending order of % missing**
3. **while** not γ **do**:
4. $X^{imp}_{old} \leftarrow$ store previous imputed matrix
5. **for** s in k **do**:
6. Fit a random forest predicting the non-missing values of X_s: $y^{(s)}_{obs} \sim x^{(s)}_{obs}$
7. Use this to predict the missing values of X_s: predict $y^{(s)}_{mis}$ using $x^{(s)}_{mis}$
8. $X^{imp}_{new} \leftarrow$ update imputed matrix, using the predicted $y^{(s)}_{misy}$
9. **end for**
10. update γ
11. **end while**
12. **return** the final imputed matrix X^{imp}

Algoritmo 1: pseudo codice Miss-forest

3.4 Feature Selection

La feature selection è il processo di riduzione del numero di variabili di input durante lo sviluppo di un modello predittivo.

È spesso opportuno ridurre il numero di variabili di input per due motivi:

1) per ridurre il costo computazionale della modellazione,
2) in alcuni casi, per migliorare le prestazioni del modello stesso.

I metodi di feature selection implicano la valutazione della relazione tra ciascuna variabile di input e la variabile target. Fatto ciò verrà utilizzato quel subset di variabili di input che hanno una relazione più forte con la variabile target.

Gli algoritmi di feature selection possono essere ampiamente classificati in due categorie:

- *minimal-optimal (come MRMR),*
- *all-relevant (come Boruta)*

I metodi minimal-optimal cercano di identificare un piccolo insieme di caratteristiche che, messe insieme, hanno il massimo potere predittivo possibile. D'altra parte, gli algoritmi all-relevant sono progettati per selezionare tutte le feature che, individualmente, hanno un potere predittivo.

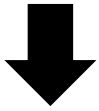
3.4.1 Boruta

Boruta è un algoritmo progettato per risolvere problemi “all-relevant” ovvero trovare un sottoinsieme di caratteristiche dal dataset che sono individualmente rilevanti per una data classificazione.

Con Boruta, le feature non competono tra loro ma con una versione randomizzata di loro stesse: In pratica, a partire dal dataset X, viene creato un altro dataset mescolando in modo casuale ogni feature. Queste feature permutate sono chiamate "shadow features". A questo punto, il “dataset shadow” è collegato al dataframe originale per ottenere un nuovo dataframe che ha il doppio del numero di colonne del dataset originario X. Le *tabelle 11 e 12* illustano il passaggio dal dataset originale ad datest shadow.

Age	Height	Weight	Income
25	182	75	20
32	176	71	32
47	174	78	45
51	168	72	55
62	181	86	61

Tabella 11



age	height	weight	Shadow_age	Shadow_height	Shadow_weight
25	182	75	51	176	75
32	176	71	32	182	71
47	174	78	47	168	78
51	168	72	25	181	72
62	181	86	62	174	86

Tabella 12

Dopo di che, una Random Forest viene addestrata sul nuovo dataset per valutare l'importanza delle singole feature. Ora analizziamo l'importanza di ciascuna feature originale e la confrontiamo con una soglia che è definita come la massima importanza della feature registrata tra le "shadow features". Quando l'importanza di una feature è superiore a questa soglia, viene chiamata "hit".

L'idea è che una feature è utile solo se è in grado di fare meglio della feature randomizzata più importante.

Presa una feature e assumendo di non avere assolutamente idea se essa sia utile o meno, la probabilità che risulti "importante" è espressa da una probabilità del 50%, come la probabilità che lanciando una moneta (bilanciata) esca testa. Poiché ogni esperimento indipendente può fornire un risultato binario (positivo o negativo), una serie di n prove segue una distribuzione binomiale.

Riportando il grafico della distribuzione binomiale in *Figura 2* possiamo individuare tre macro-aree:

- un'area di rifiuto (la zona rossa): le caratteristiche che finiscono qui sono considerate rumore, quindi vengono eliminate;
- un'area di irresolutezza (la zona blu): Boruta è indeciso sulle caratteristiche che sono in questa zona;
- un'area di accettazione (l'area verde): le caratteristiche che sono qui sono considerate predittive, quindi vengono mantenute.

  [7]

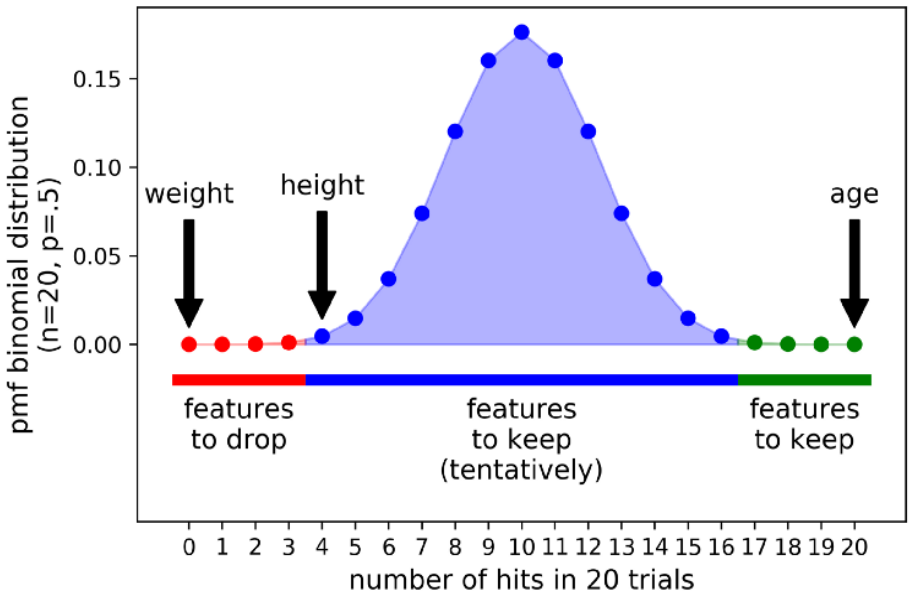


Figura 2: grafico della distribuzione binomiale diviso in tre macro-aree: area rossa le feature non vengono scelte, blu scelte ma con incertezza, verde le feature sono considerate fondamentali. Figura presa da [7]

3.4.2 MRMR

MRMR (chè è un metodo minimal optimal) cerca di identificare un insieme di caratteristiche che nella loro totalità e non nella singolarità abbiano il massimo potere predittivo possibile.

Infatti se due feature sono entrambe rilevanti ma portano più o meno le stesse informazioni, un metodo all-relevant come Boruta le seleziona entrambe, mentre un metodo minimal-optimal come MRMR seleziona solo una di esse e scarterà l'altra.

Quando si utilizza MRMR, è fondamentalmente necessario fare una sola scelta: decidere il numero di feature che si desidera mantenere. Questo algoritmo funziona in modo iterativo: Ad ogni iterazione, identifica la feature migliore (secondo una regola) e la aggiunge all'insieme di quelle selezionate. Una volta che una caratteristica viene selezionata, non può essere più scartata.

Quale è la regola che determina la scelta della caratteristica "Migliore" ad ogni iterazione? "Maximum Relevance - Minimum Redundancy": Ad ogni iterazione - si vuole selezionare la caratteristica che ha la massima rilevanza rispetto alla variabile target e la minima ridondanza rispetto alle caratteristiche che sono state già selezionate nelle iterazioni precedenti. [8]

3.5 Autoencoder

L'autoencoder è un tipo di rete neurale che può essere utilizzata per apprendere una rappresentazione compressa di dati originali.

Un autoencoder ha due parti principali (come possiamo osservare nella *Figura 3*): un encoder che mappa l'input nel codice e un decoder che associa il codice a una ricostruzione dell'input. Il modo più semplice per eseguire perfettamente l'attività di copiatura sarebbe duplicare il segnale. Invece, gli autoencoder sono tipicamente costretti a ricostruire approssimativamente l'input, preservando solo gli aspetti più rilevanti dei dati nella copia.

Esistono molti tipi di autoencoder e il loro utilizzo varia, ma l'uso più comune è come modello di feature extraction. In questo caso, una volta che il modello è stato addestrato, la parte di decoder del modello può essere scartata e può essere utilizzato il modello fino al bottleneck, ovvero fino al punto in cui la compressione dei dati è massima (appunto un collo di bottiglia). Questa parte del modello è l'encoder. L'output dell'encoder è un vettore a lunghezza fissa che fornisce una rappresentazione compressa dei dati di input.

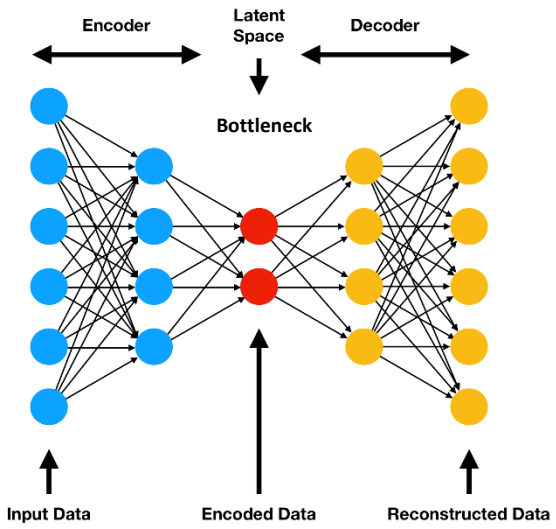


Figura 3: rappresentazione di un autoencoder. Sono evidenziate le aree principali del modello: encoder, bottleneck e decoder.

L'encoder può quindi essere utilizzato come tecnica di preparazione dei dati per eseguire feature extraction su dati grezzi che possono essere utilizzati per addestrare un modello di apprendimento automatico diverso. [9]

3.6 Differenza tra Feature Selection e Feature Extraction

La feature extarction consiste nel creare un nuovo set di feature più piccolo che acquisisce ancora la maggior parte delle informazioni utili delle feature originali. La feature selection invece è il processo di selezione di un sottoinsieme di caratteristiche rilevanti (feature) da utilizzare nella costruzione di un modello preditivo. La differenza la si può notare in maniera molto semplice dalla *Figura 4*.

Figura 4: Rappresentazione grafica della differenza tra feature selection e extraction. Nella feature selection si riduce semplicemente il numero di feature mentre nella feature extarction si passa a un nuovo set di feature (x→ y)

3.7 Parametri di valutazione di un modello di Machine Learning

Per la valutazione di un modello di machine learning sono state utilizzate diverse metriche quali accuratezza, F1 score, ROC AUC, precision e recall. Prima di spiegare queste metriche è necessario un introduzione ai concetti di "vero positivo", "vero negativo", "falso positivo" e "falso negativo". Un vero positivo è un risultato in cui il modello prevede correttamente la classe positiva. Allo stesso modo, un vero negativo è un risultato in cui il modello prevede correttamente la classe negativa. Un falso positivo è un risultato in cui il modello prevede in modo errato la classe positiva. E un falso negativo è un risultato in cui il modello prevede in modo errato la classe negativa.

Accuratezza

L'accuratezza, informalmente, è la frazione di previsioni che il modello in oggetto ha ottenuto correttamente. Formalmente, l'accuratezza ha la seguente definizione:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Per la classificazione binaria, l'accuratezza può essere calcolata anche in termini di positivi e negativi come segue:

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Dove tp = veri positivi, tn = veri negativi, fp = falsi positivi e fn = falsi negativi.

Precisione

La Precisione è il rapporto tra i veri positivi e tutti i positivi (corretti e non). La precisione, in inglese *precision*, è l'accuratezza con cui il sistema di machine learning prevede le classi positive. Si calcola con una semplice frazione:

$$\text{Precision} = \frac{tp}{tp + fp}$$

Recall (o recupero)

La recall è il rapporto tra i veri positivi e il totale di tutti i test che sarebbero dovuti essere positivi. È utile per indicare il rapporto di istanze positive correttamente individuate dal sistema di machine learning. Anche questa misura si esprime come una semplice frazione:

$$\text{Recall} = \frac{tp}{tp + fn}$$

F1 score

Spesso è conveniente fondere ***Precision e Recall*** in una sola metrica chiamata: **F_1 score**.F1 Score è una media armonica cioè il reciproco della media aritmetica dei reciproci.. Può assumere valori compresi fra 0 e 1: assume valore 0 solo se almeno uno dei due tra precisione e recall vale 0 mentre assume valore 1 se sia precisione che recupero valgono 1.

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}.$$

ROC AUC

AUC significa area sotto la curva ROC, quindi per parlare del punteggio ROC AUC dobbiamo prima definire la curva ROC. È un grafico che visualizza il compromesso tra tasso di veri positivi (TPR) e tasso di falsi positivi (FPR). Il tasso di falsi positivi viene calcolato come fp/fp+tn, dove fp è il numero di falsi positivi e tn è il numero di veri negativi (fp+tn è il numero totale di negativi). Il tasso di veri positivi è anche conosciuto come sensitività o recall, già vista in precedenza.

Fondamentalmente, per ogni soglia, calcoliamo TPR e FPR e lo tracciamo su un grafico cartesiano avente come asse delle x la FPR e come asse delle y le TPR. Possiamo vedere un esempio in Figura 5.

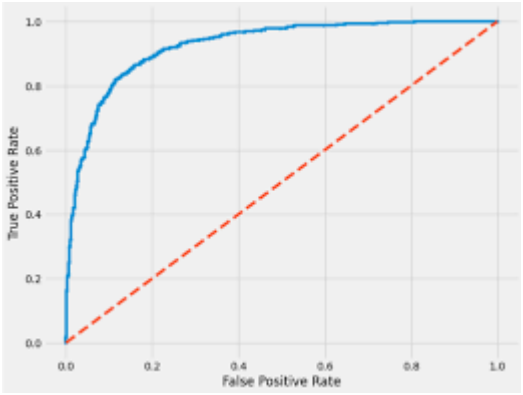


Figura 5: esempio di una curva ROC

Ovviamente, più il TPR è alto e più l'FPR è basso per ciascuna soglia, meglio è, quindi i classificatori che hanno curve più in alto a sinistra sono migliori. Per ottenere un numero che ci dica quanto è buona la nostra curva, possiamo calcolare l'area sotto la curva ROC, o punteggio ROC AUC. [10]

L'area sotto la curva come si può intuire dalla figura 5 va da 0 a 1 (per una questione matematica: l'area massima che si potrebbe ottenere sarebbe quella di un quadrato avente lato di dimensione 1). Più l'area è grande, meglio è, perchè vorrebbe dire che la curva ha raggiunto il punto con coordinate (0,1) ovvero il punto con TPR a 1 e FPR a 0.

3.8 Cross Validation

La convalida incrociata è una procedura di ricampionamento utilizzata per valutare i modelli di machine learning su un campione di dati limitato.

Si deve solo specificare un unico parametro k che rappresenta il numero di gruppi (chiamati anche fold) in cui un dato campione di dati deve essere suddiviso. Questi gruppi devono contenere approssimativamente lo stesso numero di esempi.

L'algoritmo della tecnica k-Fold (visibile alla Figura 6) funziona come segue:

1) Viene scelto un numero di fold (k). Di solito, k è 5 o 10,
2) Il set di dati viene partizionato in k parti uguali (se possibile),
3) k – 1 fold costituiranno il training set. Il gruppo rimanente sarà il test set,
4) Viene addestrato il modello sul training set. Ad ogni iterazione della convalida incrociata, è necessario addestrare un nuovo modello indipendentemente dal modello addestrato aull'iterazione precedente,
5) Viene effettuata la convalida sul test set,
6) Si salva il risultato della convalida,
7) I passaggi da 3 a 6 vengono ripetuti k volte (ogni volta il fold che costituisce il test set sarà diverso). In questo modo si avrà convalidato il modello su ogni fold del dataset,
8) Per ottenere il punteggio finale viene effettuata la media dei risultati ottenuti al passaggio 6.

Questo metodo permette di dividere il campione in modo tale che ogni fold abbia una buona rappresentazione dell'intero set di dati. Quello a cui si è interessati in particolare è che i dati delle classi esistenti siano divisi equamente nei diversi fold. [11]

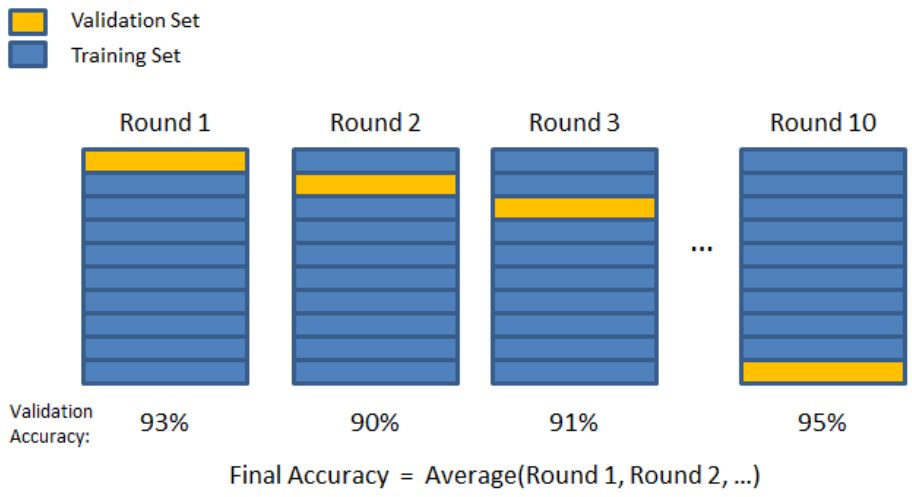


Figura 6: esempio di k-fold Cross Validation dove k=10.

4 Pratica

4.1 Dataset

4.1.1 Introduzione al Dataset utilizzato

Il dataset utilizzato per questo studio comprende 1.521 individui, di cui 1.126 dell'Union Hospital e 395 dall'ospedale di Liyuan. Tutti i pazienti erano caratterizzati da clinical feature (CF) e 1.342 di questi soggetti avevano dati sia CT che CF.

I 1.521 individui sono stati classificati in base alla positività alla malattia COVID-19. 894 sono i casi di positività al COVID-19 mentre 328 sono i casi di negativi e 299 i casi sospetti. In un secondo momento i positivi al COVID-19 sono stati classificati in tre categorie in base alla gravità della malattia e questi sono i risultati:

1. 24 casi di malattia lieve (2.7%),
2. 596 casi di malattia regolare (66.7%),
3. 202 casi di malattia grave (22.6%),
4. 72 casi classificati come critici (8.1%),

Il dataset comprende 756 individui di sesso maschile e 765 individui di sesso femminile la cui età è compresa tra 1 anno di vita fino ai 97.

Il dataset sulla quale è stato effettuato il lavoro è composto da:

1) 2 labels, rispettivamente la Mortality outcome e la positività/negatività del paziente alla malattia Covid.19,
2) 3 attributi “generali” riscontrate al momento dell’opedalizzazione del paziente (Genere, Età e temperatura corporea)
3) 167 attributi riguardanti malattie pregresse del paziente,
4) 39 attributi derivanti da analisi del paziente condotte in laboratorio (es: CO2 value, APTT value, …). Sono quelle che più avanti chiamerò clinical feature,
5) 39 attributi che indicano se i valori provenienti dalle analisi in laboratorio sono nella media (0), sopra la media (1) o sotto essa (-1).

Il dataset appena presentato conteneva molti valori mancanti, perciò è stata fatta una analisi e si è scelto di eliminare tutte quelle variabili che avevano al loro interno un numero troppo elevato di valori mancanti. Questa soglia è rappresentata dal numero 600 (40% delle righe del dataset): tutte le variabili con >=600 valori mancanti sono state eliminate dal dataset. È stato fatto ciò perché se avessimo tenuto anche quelle colonne aventi troppi valori mancanti, l’algoritmo di imputazione dei valori mancanti avrebbe lavorato su dati poco affidabili e di conseguenza avrebbe prodotto risultati altrettanto inaffidabili.

4.1.2 Svolgimento della Miss Data Imputation

Prima di effettuare la vera e propria imputazione dei valori mancanti viene analizzato il dataset con l'obiettivo di trovare le variabili categoriche del nostro dataframe. Questo procedimento è essenziale in quanto la funzione fit_transform di MissForest ha necessità di sapere quali attributi sono categorici e quali no. Una volta trovate, possiamo procedere all'imputazione che avviene nella seguente maniera: per 100 iterazioni vengono calcolati i valori mancanti del dataset (X_imputed). Inoltre dalla seconda iterazione in poi vengono calcolate anche le varianze di ogni valore mancante fino all'iterazione corrente e viene poi salvata la Between Variance (media delle varianze fino all'iterazione corrente.)

Una volta terminata l'ultima iterazione, un nuovo dataframe (avente le stesse colonne di quello originale) viene popolato con i dati imputati all'ultima iterazione.

Questo procedimento è stato ripetuto due volte: una volta con l'attributo "decreasing=True" di MissForest e una volta con "decreasing=False". Questo attributo, se impostato su True, fa sì che le colonne vengono ordinate in base al numero decrescente di valori mancanti. In altre parole, l'imputazione partirà dalle colonne con il maggior numero di valori mancanti alle colonne con minor numero di valori mancanti.

Una volta eseguito l'algoritmo con queste due impostazioni di Miss Forest, sono state analizzate visivamente le Between Variances calcolate nei due casi e questo è il risultato:

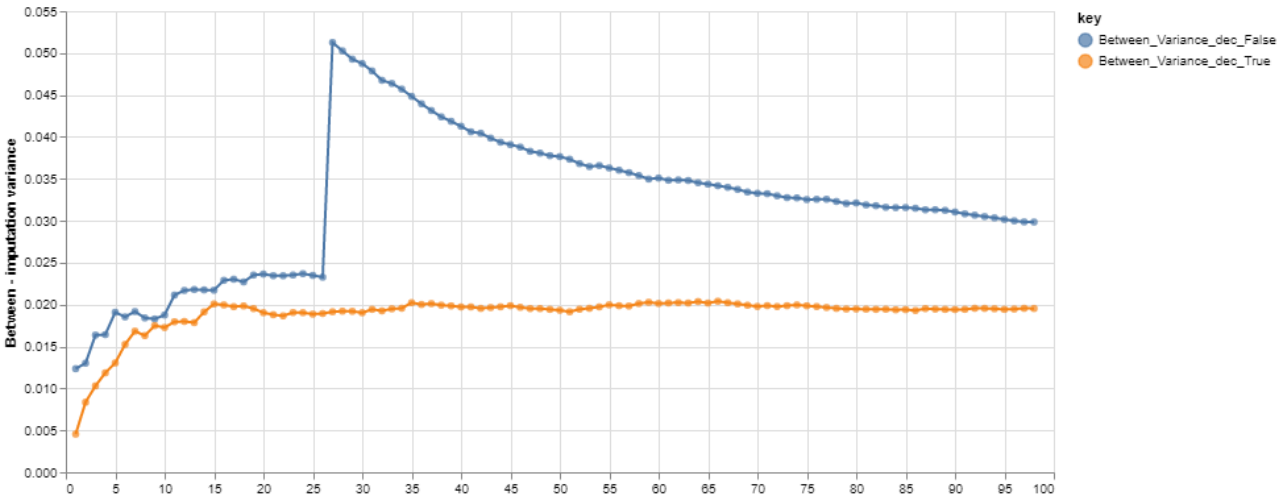


Figura 7: Between variance dell'algoritmo 'Miss Forest' (utilizzato con parametri decreasing=True/False)

Come possiamo vedere dalla *Figura 7*, la Between Variance quando l'attributo decreasing è impostato a True è inferiore rispetto a quando è impostato a False. Di conseguenza si è scelto di utilizzare il dataset prodotto dall'esecuzione di Miss forest con "decreasing=True".

4.2 Feature Selection

4.2.1 Introduzione al Dataset PRE-FEATURE SELECTION

Inizialmente, prima dell'applicazione di algoritmi di feature selection, si è scelto di dividere il dataset in tre sotto dataset:

1) Un primo dataset ("NEW_D1_malattie_sex") contenete la feature "Gender" più tutte le feature riguardanti le possibili malattie pregresse del paziente (Syncope, Aphasia, …) più gli organi colpiti da queste malattie pregresse. Questo ultimo gruppo di feature è stato aggiunto in un secondo momento al dataset originale
2) Un secondo dataset ("D2_clinical_temp_age") contenete l'età (normalizzata) più tutte le clinical feature (CO2 value,…), ovvero tutte le feature di tipo continuo del dataset originale
3) Un terzo e ultimo dataset ("D3_range_temp") contenete sempre la temperatura normalizzata (0 o 1) più tutte quelle feature che indicavano se una clinical feature era in range o meno.

La divisione in tre sotto-dataset è stata effettuata perché così facendo è stato più facile comprendere quali feature fossero ritenute importanti all'interno dei macro-gruppi (malattie pregresse, clinical feature e clinical feature "in range")

Nei paragrafi successivi è stato descritto come è stato eseguito da me il lavoro inerente alla feature selection e i suoi risultati, partendo da una prima spiegazione circoscritta al funzionamento del solo Boruta sul primo dei tre dataset sopra elencati fino ad arrivare all'esecuzione sia di Boruta che di MRMR sul dataset originale.

4.2.2 Primo approccio all'algoritmo Boruta

Inizialmente è stato applicato solamente l'algoritmo Boruta al primo dei tre dataset sopra indicati.

Per prima cosa è stata specificata una delle due label del dataset trattato come target della feature selection: infatti la selezione delle feature cambia a seconda della label di target come vedremo più avanti. Scelta la target-label, possiamo procedere con l'esecuzione di Boruta. Come spesso accade nell'apprendimento automatico, la chiave è il numero di iterazioni che vengono eseguite. Per questo motivo, Boruta in questo primo utilizzo è stato eseguito 150 volte: un for esterno da 15 iterazioni con all'interno una 10-fold cross validation. All'interno di questi due for, ad ogni iterazione, vengono scelte un numero arbitrario di feature da Boruta. Alla fine della 150-esima iterazione ogni feature sarà associata ad un numero (compreso tra 0 e 150) che indica quante volte quella feature è stata ritenuta importante per la predizione della target-label scelta. Vediamo in *Tabella 13* uno screenshot di una parte dell'output.

Feature	Score (su 150 iterazioni)
Gender	150
Dementia	134
Brain surgery	21
Brain stem hemorrhage	0
Cerebral atherosclerosis	0

Tabella 13: tabella che riassume i risultati dell'esecuzione dell'algoritmo Boruta per le 5 feature riportate. Lo score indica quante volete la feature in questione è stata selezionata da Boruta

Come possiamo vedere nella Tabella 13, la feature "Gender" è stata selezionata ben 150 volte su 150 possibili. Questo ci dice che essa è ritenuta molto importante per la previsione della target-label (in questo caso "Label1-Mortality outcome"). Lo stesso vale per "Dementia" scelta 140 volte su 150. Un discorso diverso invece lo dobbiamo fare per le altre tre feature: "Brain surgery" e "Brain stem hemorrhage" non vengono mai scelte da Boruta mentre "Cerebral atherosclerosis" viene scelta solo 21 volte, non sufficienti a superare la soglia da noi imposta del 60% (90/150).

4.2.3 Risultati di Boruta sul dataset "NEW_D1_malattie_sex"

Nella *Figura 8* vengono illustrati risultati dell'applicazione di Boruta al dataset "NEW_D1_malattie_sex":

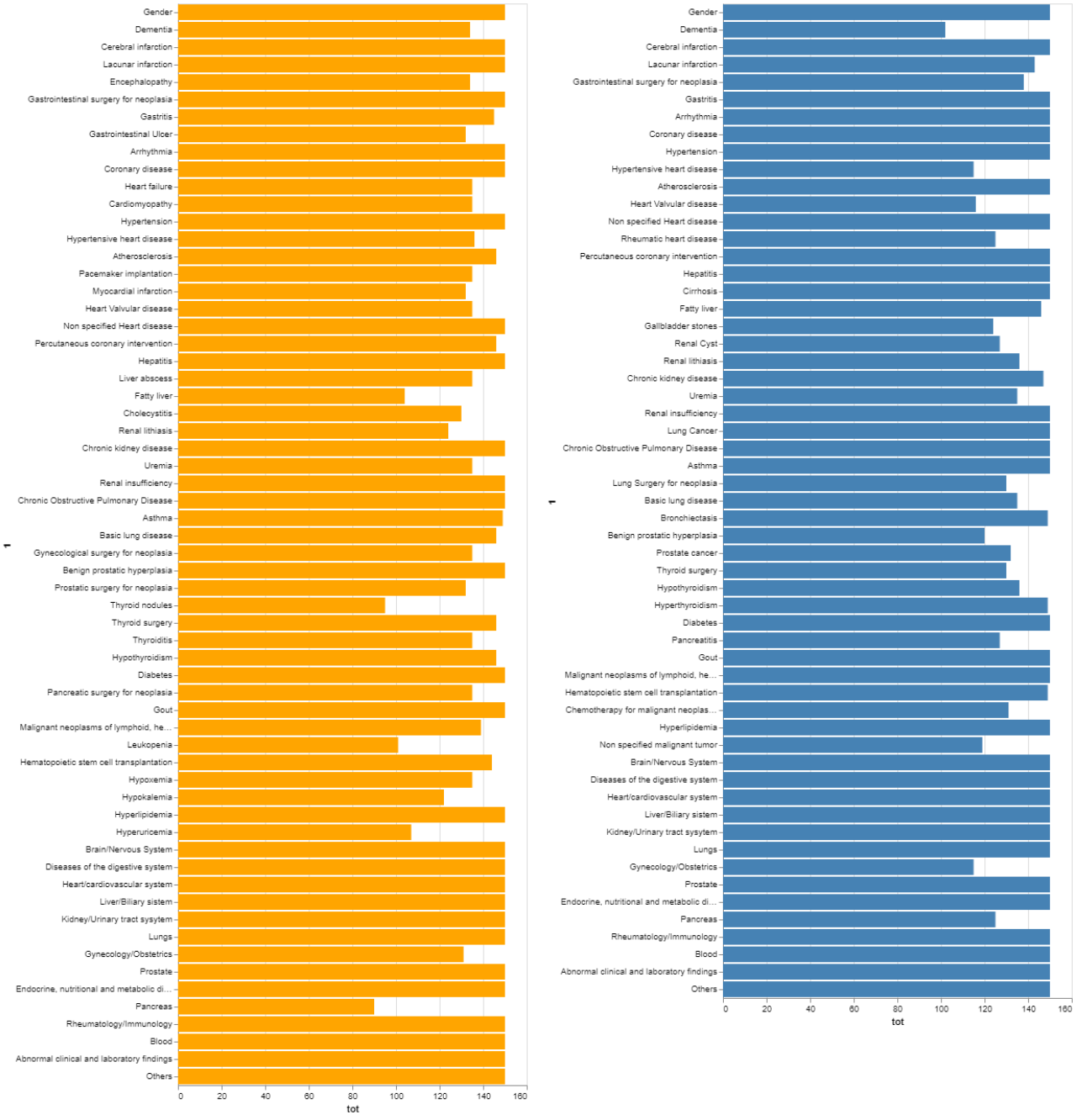


Figura 8: risultati di Boruta sul dataset "NEW_D1_malattie_sex"

Si possono osservare sempre nella Figura 8 sulla sinistra tutte le feature selezionate da Boruta per la predizione della label "Label1-Mortality outcome" mentre sulla destra quelle selezionate per la label "Label2-SARS-CoV-2 nucleic acids". Come possiamo vedere, molte feature sono state selezionate per entrambe le label mentre alcune come "Gastrointestinal Ulcer" sono state selezionate solo per la label "Label1-Mortality outcome" e altre ancora solo per la label 2.

Il lavoro svolto sul dataset “NEW_D1_malattie_sex” è stato eseguito anche per gli altri due dataset.

4.2.4 Utilizzo di MRMR

Dopo il lavoro svolto sui tre dataset trattati nei paragrafi precedenti si è tornati a lavorare sul dataset originale. A questo punto, oltre all’utilizzo di Boruta è stato anche implementato l’utilizzo di un altro algoritmo di feature selection: MRMR.

Oltre all’osservazione delle feature selezionate da Boruta e da MRMR sono state analizzate anche le feature selezionate dall’intersezione e dall’unione di questi due algoritmi.

Per utilizzare mrmr in Python bisogna installare il package pymrmr. pymRMR fornisce un unica funzione: pymrmr.mRMR. Essa richiede 3 parametri di input:

1) Il primo parametro è un DataFrame contenente il dataset di input. La prima colonna è la label di classificazione (target). Le restanti colonne sono le diverse variabili che possono essere selezionate dall'algoritmo,
2) Il secondo parametro è una stringa che definisce il metodo interno di selezione delle caratteristiche da utilizzare (definito nell'articolo originale): i valori possibili sono “MIQ” o “MID”,
3) Il terzo parametro è un numero intero che definisce il numero di caratteristiche che dovrebbero essere selezionate dall'algoritmo.

La struttura dell’algoritmo e molto simile a quella spiegata al paragrafo *4.2.2*: Viene specificata una delle due labels del dataset come target e poi per 1000 iterazioni vengono eseguiti gli algoritmi di feature selection e Boruta e MRMR. Il funzionamento è spiegato molto semplicemente dal pseudo-codice riportato in *Algoritmo 2*.

```
1. For nit in 100:
2.         Creazione n_k fold:
3.         For i in n_k fold:
4.                 Creazione punti di train e test
5.                 Calcolo set features selezionate da Boruta all’iterazione corrente
6.                 Calcolo set features selezionate da MRMR all’iterazione corrente
7.                 Calcolo set features selezionate da Boruta e/o da MRMR all’iterazione corrente
8.                 Calcolo set features selezionate da Boruta e MRMR all’iterazione corrente
```

Algoritmo 2: pseudo codice che illustra come la feature selection sia stata effettuata

Quindi, per ogni iterazione i:

- vengono selezionate un numero N arbitrario (perciò non deciso da noi) di feature da Boruta.
- Il dataset, la stringa indicante il parametro di selezione delle feature e il numero N vengono forniti come parametri alla funzione sopra citata pymrmr.mRMR,
- Le feature selezionate sia da Boruta che da MRMR vengono inserite nella lista “intesection_set”,
- Le feature selezionate da almeno uno dei due algoritmi vengono inseriti all’interno della lista “union_set”.

4.2.5 Risultati finali della Feature Selection

Una volta terminate le 1000 iterazioni complessive, i risultati sono stati salvati all'interno di un dataset comprendente 5 colonne (un estratto è riportato a Figura 9):

1) Feature: indica la feature della quale si sta trattando,
2) BORUTA: indica quante volte quella feature è stata scelta dall'algoritmo Boruta su 1000 iterazioni.
3) MRMR: indica quante volte quella feature è stata scelta dall'algoritmo MRMR su 1000 iterazioni.
4) UNION: indica quante volte quella feature è stata scelta da almeno uno tra Boruta e MRMR per ognuna delle 1000 iterazioni.
5) INTERSECT: indica quante volte quella feature è stata scelta da entrambi Boruta e MRMR per ognuna delle 1000 iterazioni.

Riporto *in Figura 9* uno screenshot del dataset contenente i risultati della feature selection.

	features	BORUTA	MRMR	UNION	INTERSECT
0	Age	1000	1000	1000	1000
1	Temb NORM	1000	1000	1000	1000
2	Gender	1000	352	1000	352
3	Dementia	0	200	200	0
4	Brain surgery	0	800	800	0
...	...	...	...	...	...
225	Rheumatology/Immunology	657	0	657	0
226	Psychiatric disorders	0	872	872	0
227	Blood	1000	1000	1000	1000
228	Abnormal clinical and laboratory findings	332	0	332	0
229	Others	1000	0	1000	0

Figura 9: risultato finale delle applicazioni dei vari algoritmi di feature selection

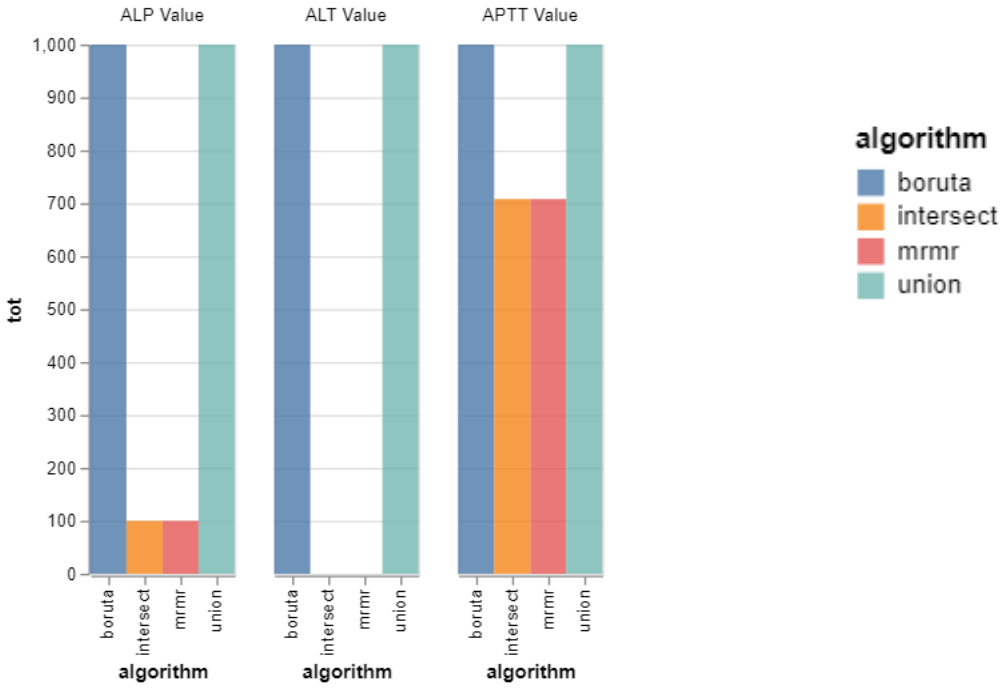


Figura 10(a): Risultati finali della feature selection per quanto riguarda le feature “ALP value”, “ALT value” e “APTT value”.

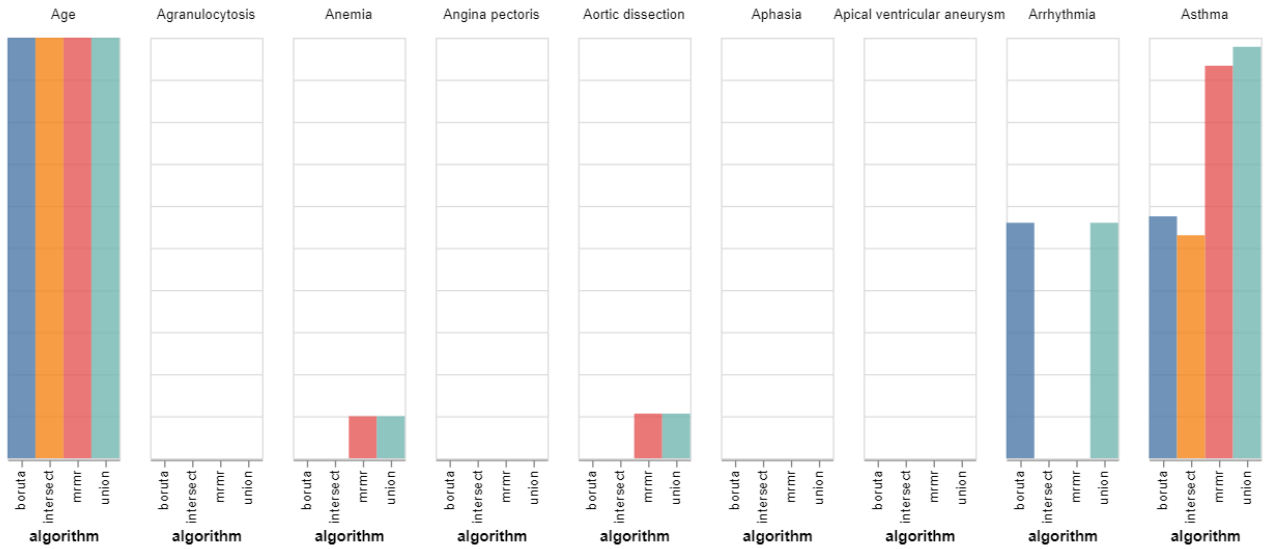


Figura 10(b): Risultati finali della feature selection riguardanti le feature “Age”, “Agranolucytosis”, “Anemia”, “Amgina pectoris”. Aortic dissection”, “Aphasia”, “Apical ventricular aneurysm”, “Arrhythmia” e “Asthma”.

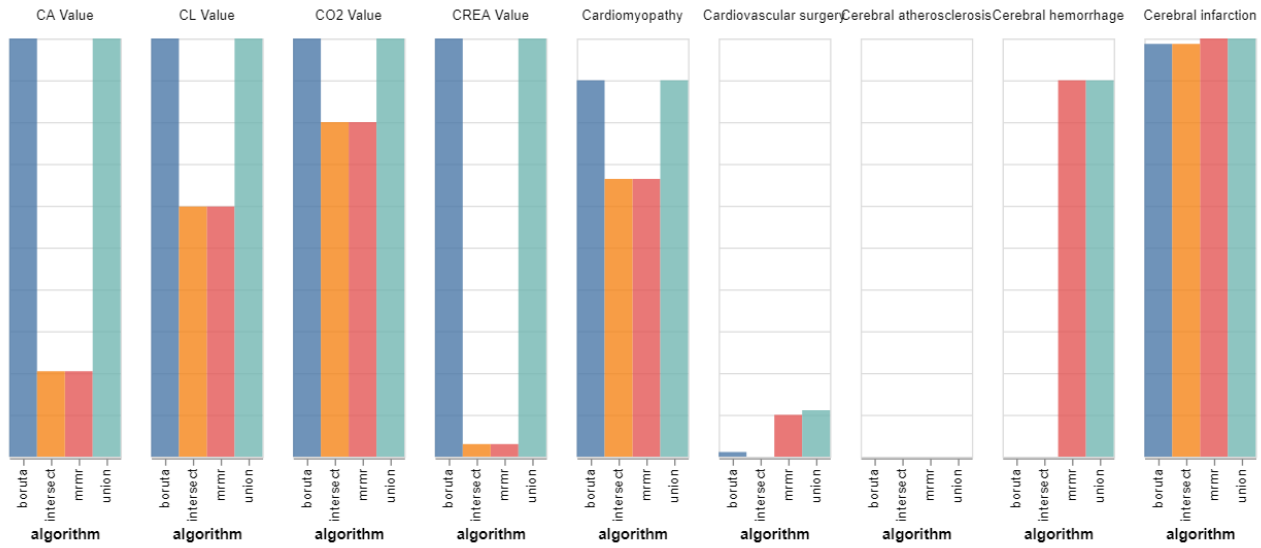


Figura 10(c): Risultati finali della feature selection riguardanti le feature “CA value”, “CL value”. “CREA value”, “Cardiomyopathy”, “Cardiovascular surgery”, “Cerebral atherosclerosis”, “Cerebral hemorrhage” e “Cerebral infraction”.

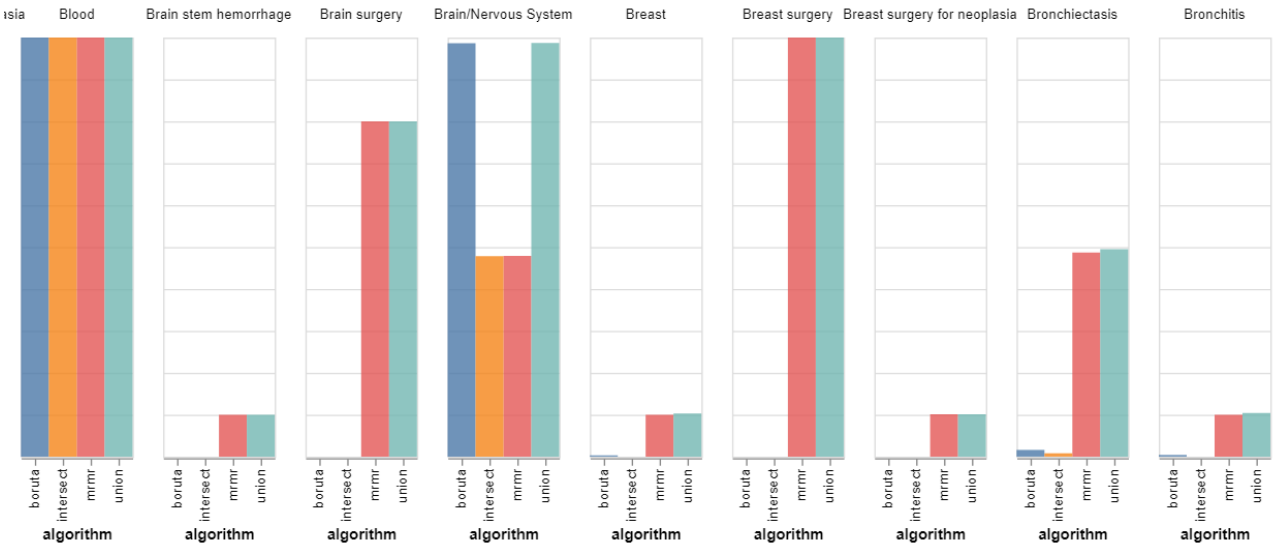


Figura 10(d): Risultati finali della feature selection riguardanti le feature “Blood”, “Brain stem hemorrhage”, “Brain surgery”, “Brain/Nervous System”, “Breast”, “Breast surgery”, Breast surgery for neoplasia”, “Bronchectasis” e “Bronchitis”.

le precedenti immagini (Figura 10(a-d)) mostrano in piccolo sotto insieme di feature e di come esse siano state selezionate dai vari algoritmi. Possiamo vedere ad esempio che la feature ‘Age’ (Figura 10(b)) è stata scelta 1000/1000 sia da Boruta che da MRMR. Di conseguenza anche la loro unione e intersezione è 1000.

4.3 Autoencoder

4.3.1 Utilizzo dell'Autoencoder

Come ultima parte di questo lavoro, è stato creato un autoencoder utilizzato per la compressione (e poi decompressione) dei dati che gli vengono passati in input. Per prima cosa il dataset è stato diviso in una parte di test e una parte di train e dopo di che il modello è stato eseguito passandogli queste divisioni del dataset come parametri.

La parte per noi più interessante di questo autoencoder è il bottleneck, ovvero il collo di bottiglia che si crea nel mezzo di questo modello nella quali i dati sono compressi al massimo. Infatti quello che è stato fatto è stato salvare il sotto modello dell'encoder e utilizzarlo per comprimere i dati in un secondo momento.

4.3.2 Encoder come preparazione dei dati per un modello predittivo

In questo paragrafo spiego come è stato utilizzato l'encoder addestrato dall'autoencoder per comprimere i dati di input e addestrare un modello predittivo diverso, in questo particolare caso una semplice rete neurale

Innanzitutto, stabiliamo una baseline delle prestazioni su questo problema. Questo è importante in quanto se le prestazioni di un modello non sono migliorate dalla codifica compressa, la codifica compressa non aggiunge valore al progetto e non dovrebbe essere utilizzata.

L'obiettivo adesso è quello di addestrare una rete neurale con i dati compressi e una rete neurale con i dati originali del dataset e vedere se si ottiene una sorte di miglioramento con i dati provenienti dall'encoder.

Questa analisi è stata fatta su diversi tipi di dataset: quello originale e quelli derivanti dalla feature selection (dataset con le feature scelte da Boruta, MRMR, …). Tutti questi esempi sono stati effettuati con l'utilizzo della k-Fold Cross-Validation (con k=3). Così facendo si evitano problemi di sovradattamento, ma anche di campionamento asimmetrico (e quindi affetto da distorsione) del campione osservato, tipico della suddivisione dei dati in due sole parti (ossia train/test).

Analizziamo il caso in cui abbiamo come dataset quello derivante dalla feature selection prodotta da Boruta: per ognuna delle tre iterazioni derivanti dalla 3-fold-cross-validation vengono creati i punti di train e test sia per i dati originali sia per i dati compressi. Dopo di che fornisco questi dati a due reti neurali e salvo i risultati in due liste. Terminata l'ultima iterazione valuto i risultati tramite classici parametri di valutazione di reti come accuracy, f1 score, precision, …

4.3.3 Risultati Autoencoder

Nelle *Tabelle 14-18* mostro i risultati relativi alle reti neurali addestrate coi i diversi dataset (Complete, Boruta, MRMR, union, intersect) nei casi in cui i dati sono stati compressi dall'encoder e non.

	accuracy	*F1score*	*precision*	*recall*	*ROC AUC*
DATASET SENZA FEATURE SELECTION	0.622	0.489	0.586	0.622	0.504
DATASET SENZA FEATURE SELECTION WITH ENCODING	0.623	0.482	0.690	0.623	0.503

Tabella 14: dati relativi al dataset completo

	accuracy	*F1score*	*precision*	*recall*	*ROC AUC*
BORUTA	0.621	0.479	0.576	0.621	0.501
BORUTA WITH ENCODING	0.621	0.476	0.386	0.621	0.500

Tabella 15: dati relativi al dataset avente le sole feature selezionate da Boruta

	accuracy	*F1score*	*precision*	*recall*	*ROC AUC*
MRMR	0.623	0.481	0.766	0.623	0.503
MRMR WITH ENCODING	0.622	0.478	0.765	0.622	0.501

Tabella 16: dati relativi al dataset avente le sole feature selezionate da MRMR

	accuracy	*F1score*	*precision*	*recall*	*ROC AUC*
UNION	0.628	0.497	0.673	0.628	0.510
UNION WITH ENCODING	0.622	0.479	0.639	0.622	0.501

Tabella 17: dati relativi al dataset avente le sole feature selezionate almeno da Boruta o da MRMR

	accuracy	*F1score*	*precision*	*recall*	*ROC AUC*
INTERSECT	0.621	0.476	0.386	0.621	0.5
INTERSECT WITH ENCODING	0.621	0.476	0.386	0.621	0.5

Tabella 18: dati relativi al dataset avente le sole feature selezionate sia da Boruta che da MRMR

4.3.4 Discussione dei risultati

Come già detto in precedenza, la compressione dei dati tramite autoencoder se non aggiunge valore al progetto, non dovrebbe essere usata, e sfortunatamente è questo il caso. Come si può leggere dalle *Tabelle 14-18* l'utilizzo di un autoencoder ha portato raramente un miglioramento di prestazioni mentre spesso ha comportato un peggioramento visibile ad esempio nella *Tabella 17.*. Solo in un caso l'utilizzo dell'autoencoder ha portato ad un miglioramento e questo è visibile nella *Tabella 14* con il parametro precision. Per quanto riguarda la feature selection, si osservano in generale lievi miglioramenti di prestazioni: ad esempio l'accuracy senza l'utilizzo di alcun algoritmo di feature selection è 0.622 mentre l'accuracy nel caso dell'unione tra feature selezionate da Boruta e MRMR è 0.628, davvero poco. In conclusione, nonostante il lavoro sia stato svolto correttamente, si può affermare che sia l'utilizzo di feature selection che l'utilizzo di un autoencoder sono stati inutili in quanto non hanno portato alcun miglioramento. Questi scarsi risultati potrebbero anche derivare da uno sbilanciamento del dataset: ad esempio su 1521 pazienti solo 57 sono pazienti morti per COVID-19 e solo 662 sono stati curati. I restanti 802 erano etichettati (nella label "Label1-Mortality outcome") come "Unknown".

5 Conclusioni

Come detto nel capitolo introduttivo, l'elaborato aveva lo scopo di sviluppare un metodo supervisionato per la predizione del rischio di mortalità per pazienti COVID-19, al fine di individuare per tempo pazienti positivi alla malattia e soprattutto capire quanto la malattia fosse grave per i singoli pazienti. Per riassumere, le principali tecnologie utilizzate sono state Miss forest, algoritmi di feature selection (Boruta e MRMR) e un autoencoder. Esse sono state impiegate in maniera sequenziale: prima è stato utilizzo Miss forest per l'imputazione dei valori mancanti, poi gli algoritmi di feature selection sono stati usati per discriminare quelle feature ritenute dai due algoritmi importanti per la predizione della target label ed infine una volta derivati nuovi dataset dalla feature selection (esisterà un nuovo dataset derivante dall'originale contenente solo le fature selezionate da Boruta, uno da MRMR ecc…), per ognuno di essi è stato implementato un autoencoder per la compressione dei dati di input.

Per ogni dataset generato, sono state create due reti neurali, una allenata con i dati del dataset (ad esempio quello derivante da Boruta) e un'altra allenata con i dati compressi (sempre di quel dataset) in modo tale da poter valutare un possibile miglioramento con i dati compressi provenienti dall'encoder (sub-model dell'autoencoder sopra citato). Per valutare quanto appena detto, sono state impiegate metriche di valutazione classica quali F1 score, recall, precison ecc…

I risultati non sono stati ottimi e si può dunque dire che gli algoritmi di feature selection e l'utilizzo di dati compressi non hanno portato alcun miglioramento dei risultati.

6 Bibliografia

[1] Machine Learning for COVID-19 Diagnosis and Prognostication: Lessons for Amplifying the Signal While Reducing the Noise by Derek Driggs, Ian Selby, Michael Roberts, Effrossyni Gkrania-Klotsas, James H. F. Rudd, Guang Yang, Judith Babar, Evis Sala, Carola-Bibiane Schönlieb on behalf of the AIX-COVNET collaboration. Articolo pubblicato su RSNA: Radiological Society of North America. URL: https://pubs.rsna.org/doi/full/10.1148/ryai.2021210011

[2] Application of Machine Learning in Diagnosis of COVID-19 Through X-Ray and CT Images: A Scoping Review by Hossein Mohammad-Rahimi, Mohadeseh Nadimi, Azadeh Ghalyanchi-Langeroudi, Mohammad Taheri and Soudeh Ghafouri-Fard. Articolo pubblicato su Frontiers Research Foundation. URL: *https://www.frontiersin.org/articles/10.3389/fcvm.2021.638011/full*

[3] Machine learning in predicting respiratory failure in patients with COVID-19 pneumonia—Challenges, strengths, and opportunities in a global health emergency by Davide Ferrari, Jovana Milic, Roberto Tonelli,Francesco Ghinelli, Marianna Meschiari, Sara Volpi, Matteo Faltoni, Giacomo Franceschi, Vittorio Iadisernia, Dina Yaacoub, Giacomo Ciusa, Erica Bacca, Carlotta Rogati, Marco Tutone, Giulia Burastero, Alessandro Raimondi, Marianna Menozzi, Erica Franceschini, Gianluca Cuomo, Luca Corradi, Gabriella Orlando, Antonella Santoro, Margherita Digaetano, Cinzia Puzzolante, Federica Carli, Vanni Borghi, Andrea Bedini, Riccardo Fantini, Luca Tabbì, Ivana Castaniere, Stefano Busani, Enrico Clini, Massimo Girardis, Mario Sarti, Andrea Cossarizza, Cristina Mussini, Federica Mandreoli, Paolo Missier and Giovanni Guaraldi. Pubblicato su National Center for Biotechnology Information URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7660476/

[4] Artificial Intelligence (AI) and Big Data for Coronavirus (COVID-19) Pandemic: A Survey on the State-of-the-Arts by QUOC-VIET PHAM, DINH C. NGUYEN, THIEN HUYNH-THE, WON-JOO HWANG AND PUBUDU N. PATHIRANA. Pubblicato da IEEE. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9141265

[5] Tony Yiu. Understanding Random Forest. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[6] *Liam Morgan.* MissForest - The best imputation algorithm. URL: *https://www.kaggle.com/lmorgan95/missforest-the-best-imputation-algorithm*

[7] Samuele Mazzanti. Boruta Explained Exactly How You Wished Someone Explained to You. 17 Marzo 2020. URL: https://towardsdatascience.com/borutaexplained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a

[8] Samuele Mazzanti. "MRMR" Explained Exactly How You Wished Someone Explained to You. URL:https://towardsdatascience.com/mrmr-explained-exactly-how-you-wished-someone-explained-to-you-9cf4ed27458b

[9] Jason Brownlee. Autoencoder Feature Extraction for Classification. URL: https://machinelearningmastery.com/autoencoder-for-classification/

[10] Metrics and scoring: quantifying the quality of predictions. URL: https://scikit-learn.org/stable/modules/model_evaluation.html

[11] Jason Brownlee. A Gentle Introduction to k-fold Cross-Validation. URL: https://machinelearningmastery.com/k-fold-cross-validation/

[12] Definizione di valore mancante. URL: https://docs.microsoft.com/it-it/analysis-services/data-mining/missing-values-analysis-services-data-mining?view=asallproducts-allversions

[13] Application of Machine Learning in Diagnosis of COVID-19 Through X-Ray and CT Images: A Scoping Review. URL: https://www.frontiersin.org/articles/10.3389/fcvm.2021.638011/full

[14] CT definizione. URL: https://it.wikipedia.org/wiki/Tomografia_computerizzata

[15] Difference Between Classification and Regression in Machine Learning. URL: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning