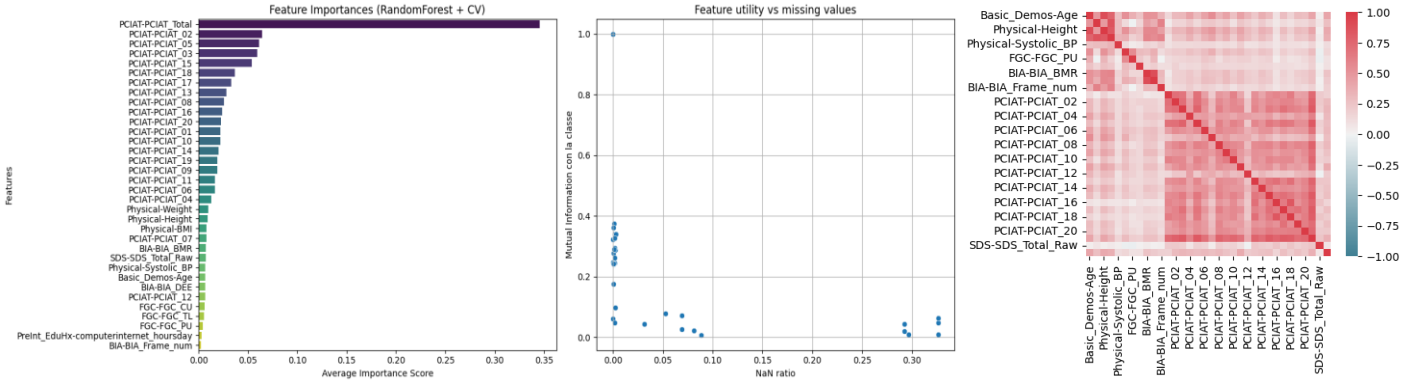
Kaggle ft Child Institute: problematic internet use

Abbiamo allenato e confrontato due modelli. uno ad albero e uno basato su rete neurale.

Preparazione Dati

Le prime cosa da fare a partire dai dati grezzi sono: split tra train/validation/test, trasformazione delle variabili categoriche in variabili numeriche e eliminazione delle features con troppi NaN, non predittive o ridondanti. Nel train abbiamo eliminato tutte le colonne con una percentuale di missing values > 50%. Abbiamo deciso di adottare un approccio ordinale per mappare le stagioni in numeri: winter:0, spring:1, summer:2, fall:3, NaN:NaN. Dopodiché abbiamo fatto la matrice di correlazione tra tutte le colonne rimanenti, eliminando le features ridondanti (correlazione > 0.9). Per scremare ulteriormente le features (per avere maggior interpretabilità e minore complessità computazionale), abbiamo allenato una random forest in cross validation, per valutare la features importance. Abbiamo anche dato un'occhiata all'analisi statistica "mutual information", che non dipende dal modello, ma misura "quanto sapere la feature aiuta a prevedere la classe". Messa in grafico col NaN ratio, dà anch'essa una buona misura di cosa sia significativo o meno, per confrontarlo coi risultati ottenuti dalla feature importance appena calcolata.

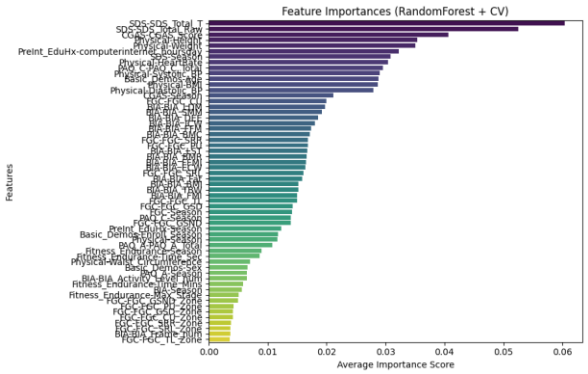


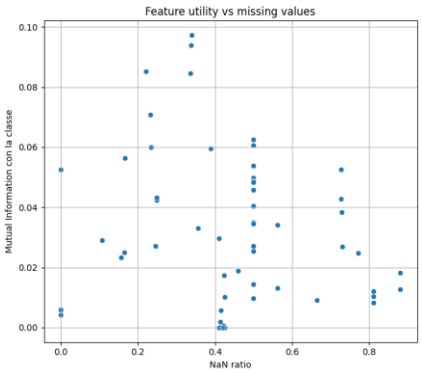
Dall'analisi della feature importance e della mutual information emergono alcune evidenze. La variabile più predittiva per l'etichetta Sii è PCIAT_Total, seguita da alcune specifiche domande (02, 05, 03, 15, 17). Le due analisi concordano: le feature più rilevanti coincidono, mentre quelle con un alto numero di valori mancanti o con bassa MI risultano poco utili e possono essere eliminate, ad esempio tramite una random forest. Tuttavia, le variabili PCIAT introducono un problema di data leakage, perché sono fortemente correlate o addirittura derivate direttamente dal target, quindi non sarebbero disponibili in scenari reali. Per questo motivo è necessario escluderle dal training del modello. Allo stesso tempo, proprio per la loro forte correlazione, esse possono essere sfruttate in un secondo momento per stimare i valori mancanti di Sii e arricchire così il dataset.

Proxy Model

Procediamo quindi con la creazione del modello di imputazione, rifacendo i passaggi già eseguiti, poiché il primo passo effettuato è stato l'eliminazione delle righe con "sii" ignoto. A partire dal dataset originale, ripetiamo i passaggi base di preparazione dei dati. In particolare, separiamo il dataset in train e test, quindi, partendo da X_train, creiamo due DataFrame distinti: uno contiene le righe del training set dove il valore di "sii" è noto, mentre l'altro include quelle in cui è NaN. Successivamente, utilizziamo le PCIAT come features e "sii" come target per allenare una semplice ma efficace random forest, che poi utilizziamo sul DataFrame con "sii" mancante per imputare i valori di "sii". Dopo aver imputato i valori mancanti, eliminiamo direttamente le colonne PCIAT, per poi ripetere la feature selection come fatto in precedenza, facendo in modo che queste colonne non vengano coinvolte nel processo. Questi i risultati: `Righe di X_training con 'sii' noto: 2193; Righe di X_training imputate: 975.`

A questo punto, procediamo a rifare i passi eseguiti in precedenza riguardanti pulizia dei dati e feature importance. (cambiare le stagioni, matrice di correlazione, feature importance, nan ratio).

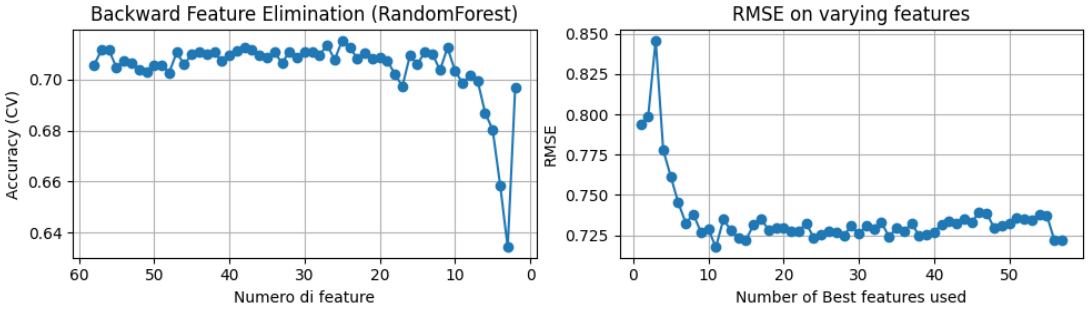




l'importance score è in generale molto più basso e decisamente meno sbilanciato di prima. Le features con NaN ratio alto non sono quelle con importance in assoluto minore, ma sono comunque nella bassa classifica, come atteso. Ugualmente alta mutual information corrisponde ad alta importance.

Feature subset selection by elimination

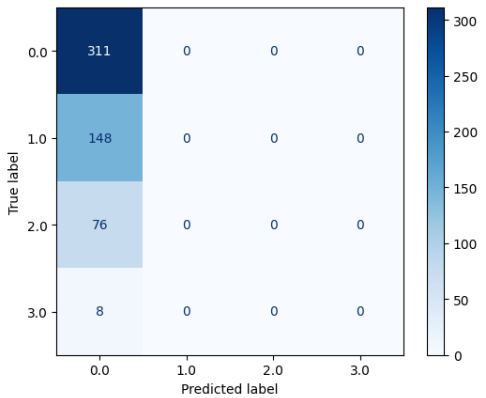
Abbiamo applicato una backward feature selection utilizzando come modello una Random Forest. Siamo partiti dal set completo di variabili e abbiamo rimosso iterativamente quelle con minor feature importance, valutando a ogni passo le performance sul validation set. Questo approccio ha il vantaggio di gestire bene le feature correlate: infatti, la Random Forest tende a dividere l'importanza tra variabili simili, ma con la rimozione progressiva una di esse può rivelarsi molto più rilevante, compensando così la penalizzazione dovuta alla ridondanza.



Da questi due grafici si evince che è possibile eliminare ripetutamente le features senza perdite in termini di accuracy: è chiaro come avere una decina di features sia tanto significativo quanto averle tutte, la perdita di accuracy è totalmente giustificata dalla semplificazione del modello. nel range 20-10 c'è una diminuzione leggermente più marcata, con maggiore varibilità, ma forse ugualmente accettabile. Da 11 in giù le performance calano sensibilmente. Abbiamo scelto di tenerne quindi 11. proseguiamo quindi con l'eliminazione di quelle non utilizzate

Modelli

Modello 0: baseline

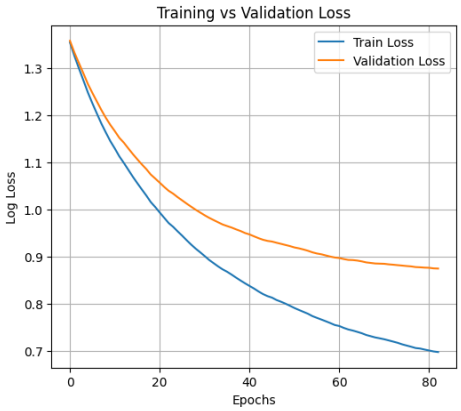


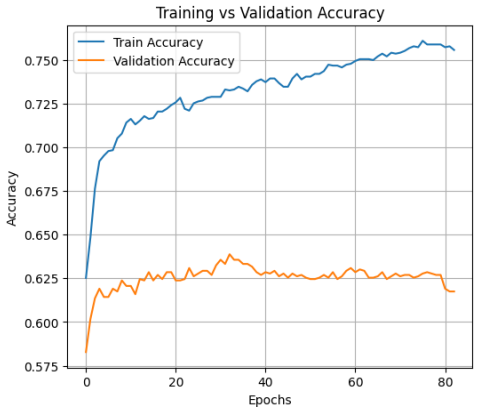
Uno dei modelli più semplici che si possono utilizzare per eseguire delle predizioni è la moda. Si nota che le classi sono fortemente sbilanciate e, essendo un test medico, è essenziale minimizzare il numero di falsi negativi della categoria 3 (corrispondente a "severe").Infatti, un falso positivo è meno problematico, in quanto dopo la predizione con il modello, si può eseguire un test reale sulla persona.In caso di un falso negativo, invece, si scarta a priori l'ipotesi ed è più improbabile che venga effettuato un ulteriore test.

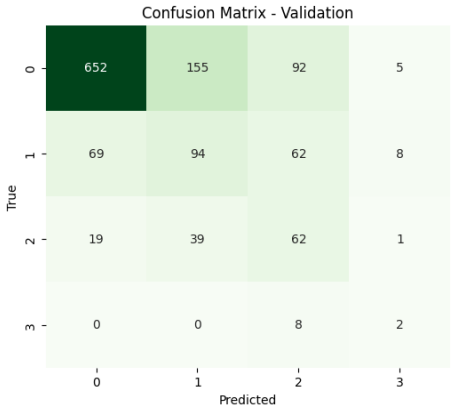
Il test **non verrà mai usato** se non alla fine, per valutare le performance dei due modelli e confrontarle con la baseline

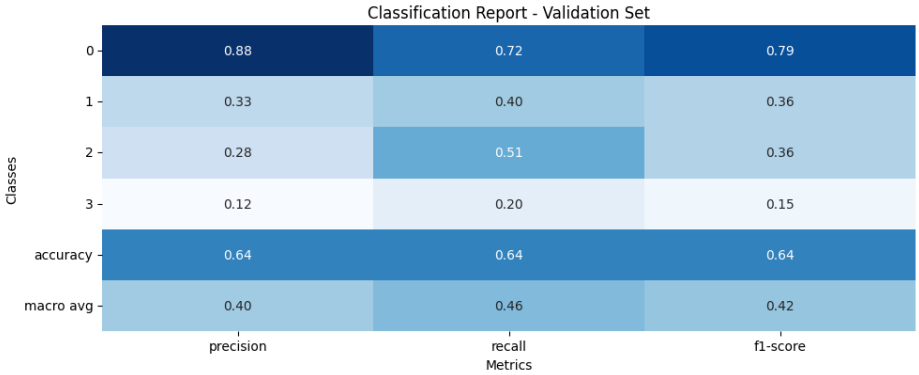
Modello 1: XGBoost

Xgboost è un algoritmo di gradient boosting ottimizato che costruisce modelli ensemble di alberi decisionali in modo molto efficiente. Nel nostro lavoro abbiamo seguito diversi passaggi per costruire e addestrare un modello di classificazione multiclasse basato su XGBoost. Per prima cosa abbiamo impostato il modello in modo da gestire più categorie, adattandolo quindi al contesto del nostro problema. Successivamente ci siamo occupati dello sbilanciamento delle classi: invece di lasciare che quelle più frequenti influenzassero troppo l'addestramento, abbiamo calcolato e applicato dei pesi proporzionati alla distribuzione. A questo punto abbiamo introdotto Optuna, una libreria che permette di automatizzare l'ottimizzazione degli iperparametri. In questo modo il modello ha potuto essere testato con diverse configurazioni e noi abbiamo potuto individuare quella che massimizzava la metrica F1 ponderata sul validation set. Infine, con gli iperparametri migliori selezionati, abbiamo addestrato il modello finale, ottenendo così una versione più robusta e meglio calibrata sulle nostre esigenze.









Modello 2: Rete Neurale, ensemble con RF

Siamo partiti dall'addestramento di una rete neurale, cercando di migliorarla variando diversi iperparametri, tra cui il numero di neuroni nei layer nascosti, le funzioni di attivazione, il dropout, la regolarizzazione L2 e il learning rate. Per affrontare il forte sbilanciamento del dataset, abbiamo anche provato a aumentare i pesi delle classi più rare. Successivamente, abbiamo testato altre strategie di miglioramento, come aggiungere neuroni o layer con regolarizzazione e utilizzare metriche alternative come MAE e MSE in virtù del rapporto ordinale tra le classi, senza però ottenere miglioramenti, anzi in particolare quest'ultima ipotesi si è rivelata totalmente fallimentare. Per la gestione dei valori mancanti, abbiamo utilizzato una prima Random Forest come imputer "intelligente", stimando i NaN tenendo conto delle relazioni tra le variabili, anziché ricorrere a media, mediana o moda.

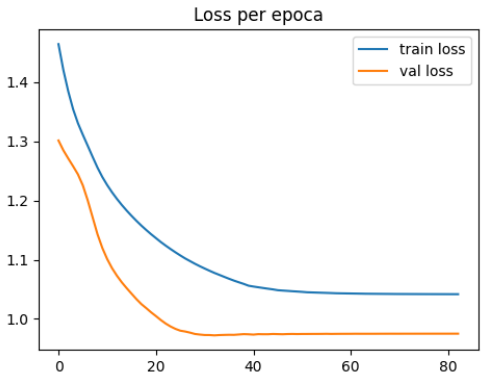
Infine, abbiamo adottato un approccio di classificazione a cascata: una seconda Random Forest viene allenata per predire la probabilità di appartenenza a due macrocategorie ("probabilmente sano - 0,1" vs "probabilmente problematico 2,3"), e la rete neurale utilizza questa nuova feature insieme alle altre per la classificazione finale. Questo approccio ha portato a un miglioramento incredibilmente significativo sulle prestazioni del modello sui dati di validazione (grafici più in basso). Questi sono i risultati della random forest. Non essendo il modello finale, ma soltanto un "aiuto", una feature in più inserita nel dataset, non è stata tunata per ottenere i migliori iperparametri.

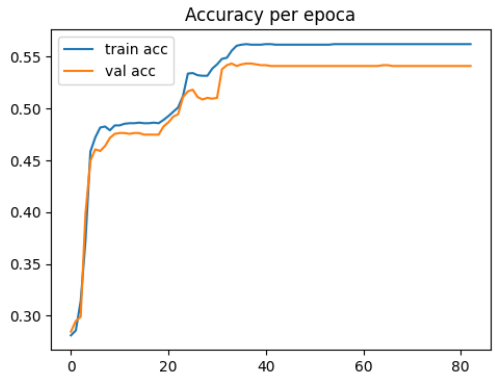
```
=== Confusion Matrix - TRAIN ===
[[1670   33]
 [   1  196]]

=== Classification Report - TRAIN ===
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      1703
           1       0.86      0.99      0.92       197

    accuracy                           0.98      1900
   macro avg       0.93      0.99      0.95      1900
weighted avg       0.98      0.98      0.98      1900

TRAIN Accuracy RF: 0.9821
```

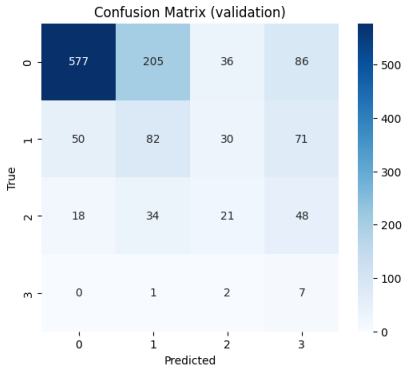
```
=== Confusion Matrix - VAL ===
[[1069   68]
 [  86   45]]

=== Classification Report - VAL ===
              precision    recall  f1-score   support

           0       0.93      0.94      0.93      1137
           1       0.40      0.34      0.37       131

    accuracy                           0.88      1268
   macro avg       0.66      0.64      0.65      1268
weighted avg       0.87      0.88      0.87      1268

VAL Accuracy RF: 0.8785
```

A questo punto, avevamo una rete neurale non molto potente, però molto migliorata con l'utilizzo della tecnica di ensemble a cascata con la random forest. Abbiamo quindi proceduto con l'hyperparameter tuning automatico sulla rete neurale, usando la libreria Keras-tuner, ottenendo risultati migliori delle aspettative. Riportiamo, per confronto, i grafici delle prestazioni tra il modello iniziale (rete neurale semplice, senza modello a cascata, senza tuning e senza imputazione dei NaN) e quello finale:

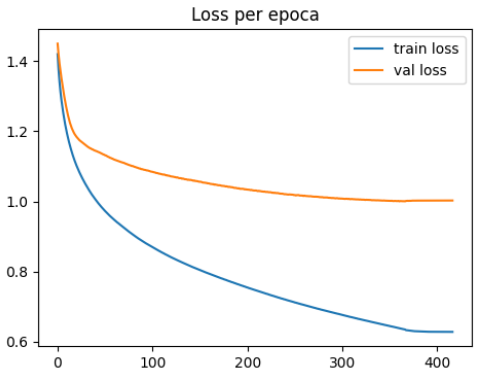
Modello semplice

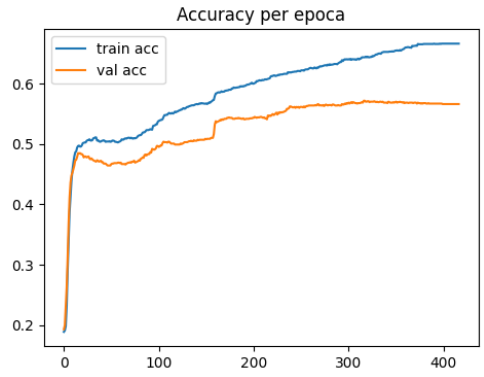


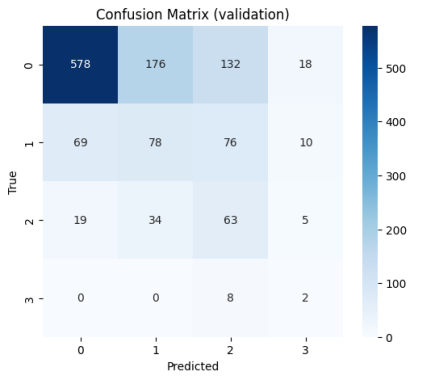




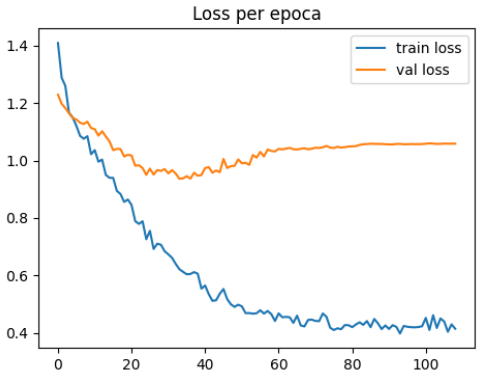
Modello con l'imputazione dei valori mancanti

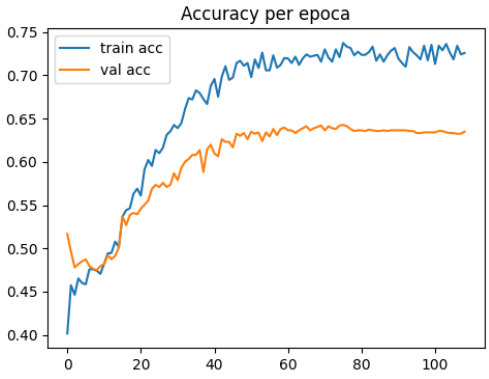


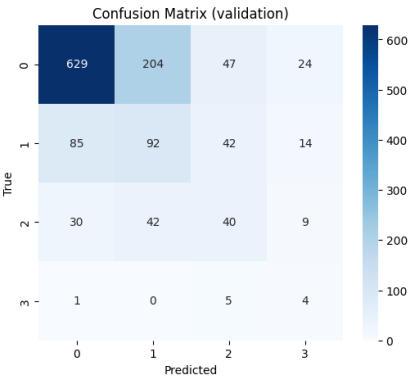




Modello con ensemble a cascata, non ottimizzato

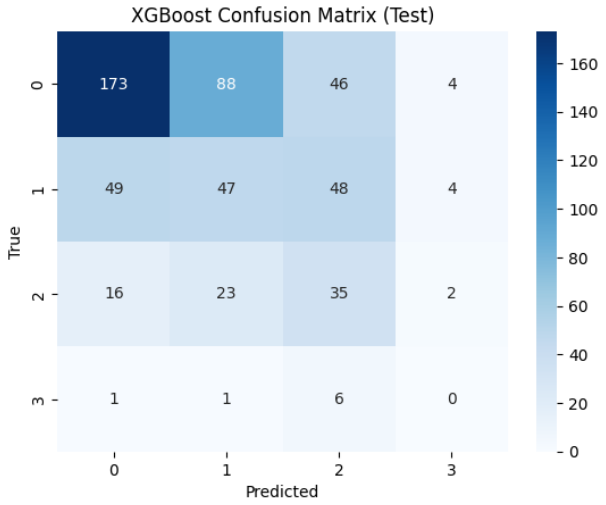






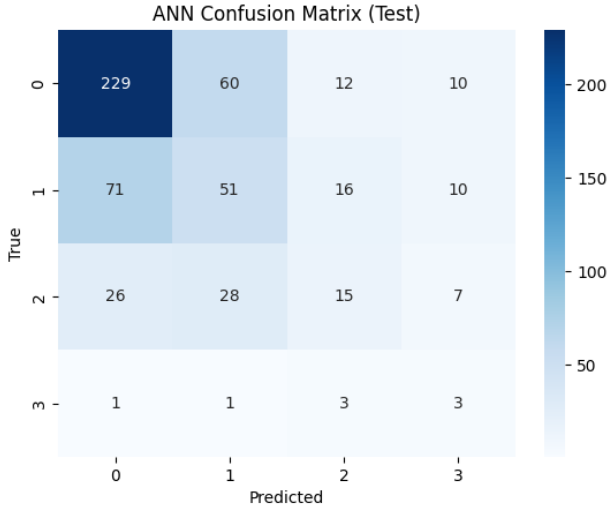
In particolare, analizzando le metriche “accuracy” e “macro avg sull’f1-score”, osserviamo sul validation set rispettivamente queste successioni, in netto miglioramento: 0.5418 -> 0.5686 -> 0.6033 e 0.3259 -> 0.3599 -> 0.3825. Il modello finale, con keras-tuner è stato scelto tra 100 modelli, ognuno dei quali allenato per 150 epoche (o meno in caso di early stopping), per un totale di 5 ore di computazione, riuscendo ad aumentare l’accuracy fino a 0.62

Test dei modelli



=== XGBoost - Classification Report (Test) ===

	precision	recall	f1-score	support
0	0.7238	0.5563	0.6291	311
1	0.2956	0.3176	0.3062	148
2	0.2593	0.4605	0.3318	76
3	0.0000	0.0000	0.0000	8
accuracy			0.4696	543
macro avg	0.3197	0.3336	0.3168	543
w.d avg	0.5314	0.4696	0.4902	543



=== ANN Ensemble - Classification Report (Test) ===

	precision	recall	f1-score	support
0	0.7003	0.7363	0.7179	311
1	0.3643	0.3446	0.3542	148
2	0.3261	0.1974	0.2459	76
3	0.1000	0.3750	0.1579	8
accuracy			0.5488	543
macro avg	0.3727	0.4133	0.3690	543
w.d avg	0.5475	0.5488	0.5444	543

Valutazione Finale

I modelli hanno un'accuracy molto modesta, in particolare è presente una certa discrepanza tra l'accuracy vista nel validation set e quella finale. In generale, XGBoost sembra che fatichi a identificare le classi minoritarie, in particolare la classe 3 ("severe"), per la quale ottiene un F1-score nullo. Al contrario, l'ensemble di Rete Neurale e Random Forest mostra una capacità superiore di generalizzazione. Riesce a classificare correttamente alcuni campioni di tutte le classi, inclusa la critica classe 3 (recall 37.5%), e ottiene un'accuracy complessiva superiore di quasi 8 punti percentuali (55% vs 47%). Un'analisi più approfondita delle confusion matrix rivela un'ulteriore aspetto di questo modello: gli errori tendono a cadere nelle classi "vicine". Ad esempio, è più probabile che un caso severo venga classificato come moderato piuttosto che come sano, indicando che il modello ha colto almeno in parte la natura ordinale del problema. Questa caratteristica, unita a una performance generale migliore (tutte le metriche -precision, recall e f1- sono migliori), rende l'ensemble preferibile per questo specifico contesto diagnostico. In entrambi i casi, un aspetto sicuramente da tenere in considerazione per la scarsa capacità predittiva dei modelli è la qualità non eccelsa dei dati iniziali e il forte sbilanciamento delle classi, che rende difficoltoso predire quelle minoritarie.