Sviluppo e Valutazione di Modelli per la Predizione di end-stage kidney disease (ESKD) in IgAN

Obiettivo Replicare e adattare un *tool* di AI per la predizione dell'**end-stage kidney disease** (ESKD) in IgA nephropathy (IgAN), seguendo un approccio a **due stadi**: (i) classificatore binario (ESKD sì/no), (ii) regressore per il *time-to-ESKD* (anni). Questo lavoro riprende e si ispira al modello proposto da **Schena et al. (2021)** nel paper *"Development and testing of an artificial intelligence tool for predicting end-stage kidney disease in patients with immunoglobulin A nephropathy"*, pubblicato su *Kidney International*.

Pulizia & feature engineering A partire da sorgenti eterogenee sono state mantenute le variabili: *Age*, *Gender*, *Hypertension*, *Proteinuria*, *Creatinine*, *MEST-C* e *Therapy*. **Hypertension** è 1 se SBP ≥ 140 o DBP ≥ 90 o terapia antipertensiva. **Therapy** è aggregata da tre indicatori (*Antihypertensive*, *Immunosuppressants*, *FishOil*). Per la classificazione viene creato anche un flag *Eskd*, che assume valore 1 se il paziente è nello stato di end-stage renal disease, 0 altrimenti. È applicata la *z-score normalization* alle feature numeriche.
Class imbalance: la percentuale di positivi ESKD è inferiore (split tipico $\approx 77\%/23\%$), quindi in training si utilizza un *WeightedRandomSampler* (PyTorch) o pesi equivalenti.

Divisione dataset e riproducibilità Si utilizza uno **split stratificato 80/20** per il test e una **10-fold cross-validation** sullo *train+val*. Dalla procedura di cross-validation si ottengono dieci modelli, ciascuno addestrato sulla propria porzione del dataset; per il test finale viene selezionato il modello che ha raggiunto il valore più alto di F1-Score sulla rispettiva porzione di validazione.
Per garantire la riproducibilità completa, tutti i generatori casuali (`torch`, `numpy` e `scikit-learn`) vengono inizializzati con un unico seed globale, impostato a `123` o `42` a seconda dell'esperimento. Ogni run utilizza quindi un solo seed per controllare l'intera pipeline (split dei dati, inizializzazione dei pesi, dropout, campionamento, ecc.), mentre l'uso di due seed distinti consente di stimare la stabilità del modello rispetto alla casualità.

Classificazione (due implementazioni):

- **PyTorch**: MLP con 4 layer nascosti da 100 unità, *BatchNorm*+ELU+Dropout; ottimizzazione Adam; *ExponentialLR* con γ calcolato per passare da LR iniziale a LR finale tra step definiti; *Proxy AUC loss* per massimizzare l'AUC-ROC; *early stopping* su validation.
- **Keras**: architettura analoga (ELU, BN/Dropout), *Proxy AUC loss*; scheduler *ExponentialDecay*; *early stopping* (con controllo della validation loss ogni 5 epoche).

Regressione (PyTorch): MLP con 3 layer da 125 unità, attivazione *SELU*, Dropout (input 0.5, hidden 0.3), ottimizzazione Adam, *ExponentialLR*, loss MSE (opz. MAE), *early stopping*. L'output è il tempo stimato (anni) fino a ESKD, $\in [0, 10]$.

Metriche Classificazione: *Accuracy*, *Precision*, *Recall*, *F1*, *AUC-ROC*.
Regressione: *MAE*, *RMSE*.

Risultati. Le seguenti tabelle riportano le metriche di classificazione per i due seed considerati e tre sottoinsiemi del test set: *all*, dateAssess ≥ 5 anni e dateAssess ≥ 10 anni. La Tabella 1 riassume invece i risultati originali riportati da Schena et al. (2021).

Table 1: Risultati di riferimento dal paper di Schena et al. (2021).

Follow-up	Accuracy	Precision	Recall	F1-score	AUC
5 anni	0.80	0.83	0.92	0.87	0.82
10 anni	0.83	0.81	0.89	0.85	0.89

In un primo esperimento, sono state considerate tutte le osservazioni disponibili, utilizzando **l'ultima visita per ciascun paziente** come riferimento temporale. I risultati sono riportati di seguito.

Table 2: Classificazione **PyTorch** (solo **ultima visita per paziente**): metriche sul **set di test**.

Test subset	Seed	Accuracy	Precision	Recall	F1	AUC
All patients	123	0.926	0.837	0.837	0.837	0.963
All patients	42	0.899	0.786	0.767	0.776	0.918
dateAssess $\geq$ 5 years	123	0.950	0.950	0.792	0.864	0.961
dateAssess $\geq$ 5 years	42	0.920	0.778	0.808	0.792	0.932
dateAssess $\geq$ 10 years	123	0.943	0.714	0.833	0.769	0.989
dateAssess $\geq$ 10 years	42	0.909	0.700	0.700	0.700	0.895

Table 3: Classificazione **PyTorch** (solo **ultima visita per paziente**): medie sui seed per ciascun subset.

Test subset	Accuracy	Precision	Recall	F1	AUC
All patients	0.913	0.811	0.802	0.807	0.941
dateAssess $\geq$ 5 years	0.935	0.864	0.800	0.828	0.946
dateAssess $\geq$ 10 years	0.926	0.707	0.767	0.735	0.942

Successivamente è stato condotto un esperimento analogo considerando **solo la prima visita disponibile per ciascun paziente**, per valutare il modello su dati non ripetuti nel tempo. I risultati corrispondenti sono riportati nelle tabelle seguenti.

Table 4: Classificazione **PyTorch** (solo **prima visita per paziente**): metriche sul **set di test**.

Test subset	Seed	Accuracy	Precision	Recall	F1	AUC
All patients	123	0.899	0.731	0.884	0.800	0.938
All patients	42	0.878	0.708	0.791	0.747	0.989
dateAssess $\geq$ 5 years	123	0.916	0.769	0.833	0.800	0.960
dateAssess $\geq$ 5 years	42	0.899	0.731	0.731	0.731	0.906
dateAssess $\geq$ 10 years	123	0.943	0.714	0.833	0.769	0.883
dateAssess $\geq$ 10 years	42	0.879	0.583	0.700	0.636	0.866

Note implementative Tutti i generatori casuali (`torch`, `numpy` e `scikit-learn`) vengono inizializzati con un unico seed globale, impostato a `123` o `42` a seconda dell'esperimento. In questo modo, ciascun run è completamente riproducibile (split dei dati, inizializzazione dei pesi, dropout, campionamento, ecc.), mentre l'uso di due seed distinti consente di valutare la stabilità del modello rispetto alla casualità. È impiegato il *WeightedRandomSampler* per bilanciare le classi, con logging tramite `wandb` e salvataggio automatico del modello con la F1 di validazione più alta. In Keras:

Table 5: Classificazione **PyTorch** (solo **prima visita per paziente**): medie sui seed per ciascun subset.

Test subset	**Accuracy**	**Precision**	**Recall**	**F1**	**AUC**
All patients	0.889	0.720	0.838	0.774	0.964
dateAssess $\geq$ 5 years	0.908	0.750	0.782	0.766	0.933
dateAssess $\geq$ 10 years	0.911	0.649	0.767	0.703	0.875

Table 6: Regressione **PyTorch**: metriche sul **set di test**.

Seed	**MAE [anni]**	**RMSE [anni]**
42	4.422	5.129
123	3.566	4.422

Table 7: Regressione **PyTorch**: media sui seed.

MAE [anni]	**RMSE [anni]**
3.994	4.776

`tf.random.set_seed(seed)`, scheduler *ExponentialDecay* ed *early stopping* a controlli periodici. Entrambe le pipeline salvano lo *scaler* e applicano il *fit* solo sui dati di training, usando il *transform* su validation e test per evitare *data leakage*.

Gli esperimenti in **PyTorch** sono stati eseguiti su **GPU NVIDIA GeForce GTX 1080 Ti**, mentre quelli in **Keras** sono stati condotti direttamente su **CPU**.

Conclusioni Il framework a due stadi (classificazione + regressione) è riproducibile e coerente con la letteratura di riferimento, in particolare con i risultati del lavoro di **Schena et al. (2021)**. L'uso di due esperimenti indipendenti con seed globali distinti (**42** e **123**) consente di garantire la riproducibilità degli esperimenti e di valutare la robustezza del modello rispetto alla casualità.

Codice sorgente Tutto il codice sviluppato, inclusi preprocessing, addestramento e analisi delle metriche, è disponibile pubblicamente al seguente indirizzo GitHub:

`https://github.com/gabri1997/prediction-eskd`

Riferimento bibliografico

Schena FP, Anelli VW, Trotta J, Di Noia T, Manno C, Tripepi G, D'Arrigo G, Chesnaye NC, Russo ML, Stangou M, *et al.* (2021). *Development and testing of an artificial intelligence tool for predicting end-stage kidney disease in patients with immunoglobulin A nephropathy.* **Kidney International, 99**(5):1179–1188.