Titolo: parkinson-analysis

Gruppo di lavoro :

- Paolo Marchisella, [MAT. 796117], p.marchisella@studenti.uniba.it

URL repo associato: https://github.com/polusy/parkinson-analysis

AA 2025-26

Introduzione

Il dominio di interesse del progetto **(parkinson-analysis)** si colloca nell'ambito dei sistemi di supporto alle decisioni cliniche, con un'attenzione specifica sulla diagnosi precoce della malattia di Parkinson attraverso l'analisi di marcatori vocali.

La patologia, essendo un disturbo neurodegenerativo, influenza significativamente il controllo motorio dei muscoli vocali [1], producendo alterazioni di features vocali (come tremori o instabilità dell'ampiezza). Il sistema presentato opera come un **sistema ibrido basato su conoscenza**: esso integra l'analisi statistica derivante dall'**Apprendimento Automatico** con una **Base di Conoscenza**, modellata su evidenze cliniche, regole complesse di inferenza e validazione/confutazione della predizione. L'obiettivo è superare i limiti dei singoli approcci, garantendo una diagnosi che sia non solo predittiva, ma anche coerente e spiegabile dal punto di vista clinico.

Sommario

Il sistema sviluppato è un **KBS (Knowledge-Based System)** progettato per la diagnosi assistita, che affronta il problema clinico attraverso un'architettura ibrida (apprendimento di modello di classificazione probabilistica in ensemble + sistema di ragionamento automatico basato su regole). La competenza multidisciplinare richiesta dal corso è dimostrata dall'integrazione di tre moduli principali: un classificatore probabilistico (**Gradient Boosting Ensemble model**), una base di conoscenza deterministica e un modulo di validazione incrociata (utilizza la conoscenza di fondo della base di conoscenza e, tramite interrogazione della stessa, analizza il risultato della predizione del modello probabilistico). Il KBS non fornisce un output binario isolato, ma esegue un ragionamento deduttivo per confermare o smentire la predizione statistica sulla base di inferenza di features vocali (marcatori vocali) astratte, conflittualità tra dati di features, severità della diagnosi effettuata e validazione incrociata con il risultato della predizione del modello di classificazione probablistica.

Elenco argomenti di interesse

- **Apprendimento e Incertezza (Cap 7 – Apprendimento supervisionato):** La gestione dell'incertezza intrinseca nelle features vocali (marcatori vocali) è affidata a un modello di apprendimento supervisionato, costituito da un regressore logistico iniziale e più regressori lineari in sequenza (**Ensemble Gradient Boosting**). La realizzazione del modello ha portato al confronto del modello Ensemble con un altro modello di classificazione probabilistica (Weak Learner), quale un regressore logistico, poi posto come primo modello di apprendimento "Weak" all'interno dell'Ensemble Boosting. Inoltre, l'apprendimento dei differenti modelli di classificazione probabilistica ha richiesto una preliminare fase di pre-processing del dataset utilizzato.

- **Rappresentazione della conoscenza (Cap 15 – Rappresentazione con Logica del primo ordine) :** La conoscenza medica relativa ai marcatori acustici del Parkinson è stata rappresentata utilizzando la logica dei predicati del primo ordine, specificamente attraverso le clausole di Horn. Questa scelta ha permesso di mappare le feature numeriche (come Jitter e Shimmer), presenti esplicitamente nel dataset originale, in simboli logici e relazioni semantiche. La base di conoscenza definisce in modo deterministico le condizioni di coerenza clinica, permettendo al sistema di distinguere tra evidenze deboli, medie o forti e di classificare eventuali discordanze tra il profilo acustico del paziente e la predizione del modello probabilistico.

  La KB non è utilizzata come semplice database di fatti, ma come sistema inferenziale capace di: propagare sintomi lungo una gerarchia semantica, calcolare severità pesate, rilevare incoerenze cliniche (conflittualità tra valori clinici di features), validare o smentire predizioni probabilistiche.

- **Ragionamento Automatico (Cap 15 - Ragionamento Relazionale):** Il sistema sfrutta inferenza basata sul linguaggio Prolog per eseguire il ragionamento automatico sulla base di conoscenza. Attraverso meccanismi di *unificazione* e *backtracking*, il sistema interroga la KB per validare le predizioni ottenute dal modulo di classificazione probabilistica. Il ragionamento è impiegato per: derivare sintomi impliciti a partire dalle feature, propagare relazioni gerarchiche tra sintomi, identificare errori critici (false negative, false positive), classificare la coerenza tra predizione probabilistica e profilo clinico.

  La struttura delle regole introduce priorità decisionali e regole di inferenza complesse, andando oltre il semplice pattern matching su fatti espliciti.

Requisiti Funzionali

Questo progetto è stato realizzato in Python 3.10+. L'ambiente di sviluppo utilizzato è Visual Studio Code.

Librerie utilizzate:

- **pandas e numpy**: importazione e manipolazione dei dataset .csv, operazioni numeriche varie (e.g. array transformation, exp. operator)
- **scikit-learn**: utilizzata solamente per effettuare random sampling raggruppato dei pazienti, split del dataset (training, test), e creazione degli indici dei folders di validazione e di addestramento all'interno dei metodi di k-folds-cross-validation.
- **pyswip**: per l'integrazione della KB in Prolog, all'interno dell'ambiente python, per la possibilità di consult della KB e di query di quest'ultima.

Apprendimento e incertezza

Sommario

Il modello di apprendimento automatico utilizzato per affrontare il problema clinico della predizione probabilistica di una condizione patologica di un paziente, e quindi della classificazione binaria dello stato di un paziente, è stato un **Gradient Boosting Ensemble Model.** La realizzazione del modello di apprendimento automatico, senza l'utilizzo di modelli già implementati e pronti all'uso (vedi scikit-learn), ha attraverso differenti fasi, che saranno descritte con profondità diverse in base alla sezione dell'argomento (Strumenti utilizzati, decisioni di progetto, etc…):

- Fase di **normalizzazione del dataset** e **undersampling**.
- Fase di creazione del primo ***weak learner*** dell'Ensemble Model
- Fase di creazione dei successivi ***weak learners*** e dell'Ensemble Model
- Fase di **scelta degli iper-parametri di regolarizzazione** dei due tipi di Weak Learner
- Fase di **organizzazione dell'Ensemble Model** e scelta dell'iperparametro riguardante il **numero di round di boosting** dello stesso.
- **Fase di testing** del modello Ensemble sui due diversi dataset a disposizione, per confrontare la capacità di generalizzazione e di stabilità del modello, a seconda dell'utilizzo di un dataset di training bilanciato (stessa proporzione di samples con target positivo e target negativo), o di un dataset di training non-bilanciato.

Strumenti utilizzati

Dataset, Normalizzazione e Undersampling:

- Il Dataset: Oxford Parkinson's Disease Detection Dataset (UCI)

- Il dataset, proveniente dal **UCI Machine Learning Repository** [1], **non è sintetico**, bensì è il risultato di analisi cliniche di features vocali di pazienti affetti e non affetti da Parkinson. Il dataset è composto da un insieme di **misurazioni vocali** provenienti da 31 persone, di cui 23 affette da malattia di Parkinson (PD). Contiene circa 195 istanze (record). Ogni riga corrisponde a una registrazione vocale (una media di 6 registrazioni per individuo). La colonna "status" è il target binario: **0 per i soggetti sani** e **1 per i soggetti affetti da Parkinson**.

- Per il **pre-processing** è stata utilizzata una tecnica di **normalizzazione tramite media e deviazione standard (Z-score Normalization)**. In particolare, il dataset originale presentava feature con differenze enormi in termini di ordini di grandezza, e questo avrebbe portato all'individuazione di parametri troppo piccoli o troppo grandi in base al tipo di unità di misura utilizzata dalla feature specifica. Così, è stato recuperato il dataset originale, ed eseguito uno split in training set e test set. Si fa notare che la normalizzazione sui valori delle features del dataset .csv **non è stato eseguito prima dello split**, in modo da evitare che il training set portasse con sé informazioni sulla media e sulla deviazione standard del test set, e quindi ci fosse **"Data Leakage",** ovvero esposizione di informazioni del set di test in quello del set di training. Perciò, una volta ottenuto lo split di training e test set, si è eseguita la normalizzazione sul training set (ad ogni valore di colonna specifica, è stata sottratta la

media dei valori di quella feature per quel particolare set di dati, e il tutto diviso per la deviazione standard dei valori della colonna specifica).

La normalizzazione è **avvenuta prima sul training set**, e successivamente, salvando i risultati di media e deviazione standard per ogni colonna, è avvenuta, utilizzando i risultati salvati, sul set di test, in modo che i dati di test venissero normalizzati seguendo la distribuzione del test di training. La procedura utilizzata ha prevenuto, quindi, il fenomeno del Data Leakage.

Inoltre, poiché il dataset si componeva di **gruppi di samples per ogni paziente** (circa 6 samples per paziente), **lo split doveva garantire** che **nessun sample** di un particolare **paziente potesse trovarsi in contemporanea nel set di training e nel set di test**. Possiamo immaginare, infatti, che questo avrebbe facilitato notevolmente le performance del modello sui dati di test, permettendogli, banalmente, di "ricordare" features e target simili presenti nel dataset di training. Per fare ciò è stato utilizzato uno strumento di scikit-learn **(GroupKFold o GroupShuffleSplit),** per effettuare splitting del dataset originale, **considerando dataframes row non singolarmente,** ma per gruppi di pazienti con stesso ID **(Group-Aware splitting).** In questo modo,i gruppi di ID presenti nel dataset di training e nel set di test sono insiemi disgiunti, e **non viene permesso "Data Leakage"** .

- L'**Undersampling** è stato realizzato tramite un algoritmo di campionamento casuale, utilizzando particolari funzioni messe a disposizioni da **pandas,** per la manipolazione dei dataframes. In particolare, poiché il dataset si componeva di 75% di pazienti con feature target ("status") positiva (1), questo costruiva il modello di regressione logistica e, in generale, l'ensemble model con un Bias fortemente direzionato alla predizione della classe di target positiva (1). L'undersampling è stato eseguito sui samples con target positivo, riducendo il numero di samples positivi nel dataset e, riportando, quindi, la proporzione del numero di samples positivi sul totale uguale alla proporzione del numero di samples negativi sul totale.

- Si fa notare, tuttavia, che **entrambi i dataset sono stati conservati**, per permettere un confronto della capacità di generalizzazione e stabilità del modello Ensemble e di regressione logistica, su un dataset bilanciato e su un dataset non-bilanciato.

Regressore Logistico (1° Weak Learner)

- Il regressore logistico è stato implementato *(ex-novo)* affinché potesse far parte, come primo weak learner, dell'Ensemble Model (guarda il perché nelle decisioni di progetto). La scelta è ricaduta su questo modello per la sua capacità di fornire una stima probabilistica iniziale, fondamentale per calcolare i **residui predittivi** su cui verranno addestrati i successivi componenti dell'Ensemble. E' stato implementato, guardando la letteratura scientifica, utilizzando **un predittore lineare** e **la funzione sigmoide,** per mappare il risultato "logit" in uno spazio probabilistico. Inoltre, l'ottimizzazione dei parametri della funzione lineare di base avviene per mezzo di **Mini-Batch Gradient-Descent** e **regolarizzazione L2** (si discute in seguito della decisione di progetto).

3. Weak Learners (Regressori Lineari) e Algoritmo di Ensemble (Gradient Boosting)

L'architettura di apprendimento finale è un **Ensemble Model** basato sulla tecnica del Gradient Boosting. A differenza di un approccio con alberi di decisione sequenziali **(Gradient Boosted Decision Trees)**, il sistema è stato progettato come segue:

- **Il Primo Modello:** Un Regressore Logistico completo (Combinazione lineare + Sigmoide) che fornisce la base predittiva di probabilità di assegnazione di un paziente ad una classe.

- **Modelli Successivi (Weak Learners):** Una serie di **Regressori Lineari** addestrati non direttamente sul target binario (0/1), ma sui **residui** (errori) calcolati tra il target reale e la predizione (probabilistica) ottenuta dalla somma dei logit dei modelli precedenti.

- **Integrazione dei Modelli:** Ogni nuovo weak learner tenta di minimizzare il gradiente della funzione di costo (**Mean Squared Loss** per regressori lineari), per mezzo di **Mini-Batch Gradient-Descent** e **regolarizzazione L2** . Il risultato finale del sistema non è la media delle predizioni, ma la **somma pesata dei logit** di tutti i modelli, la quale viene infine passata attraverso una funzione **Sigmoide** per ottenere la probabilità finale di patologia.

Decisioni di Progetto

- **I parametri di ogni regressore (logistico e lineare)** : Ogni regressore logistico e lineare utilizza un numero di parametri pari al **numero delle features nel dataset** originale, a cui si sottrae la feature 'name' e 'status' (feature target), e si **aggiunge un parametro bias**, indipendente da una specifica feature del dataset.

- **Scelta del tipo di discesa del gradiente** : si è scelto di implementare un training, sia del modello lineare che logistico, per mezzo di **Mini-Batch Gradient Descent**. Si è giunti a questa scelta come compromesso tra un **Batch Gradient-Descent** (aggiornamento dei parametri tramite media del gradiente, calcolato su ogni esempio del training set/unica batch) e uno **Stochastic Gradient-Descent** (aggiornamento immediato dei parametri tramite sottrazione del gradiente, dopo il calcolo del gradiente della funzione di perdita dopo ogni esempio del training set).

- **Scelta delle metriche di loss per la discesa del gradiente** : le metriche di loss utilizzate durante la discesa di gradiente sono state differenti, in base al tipo di regressore usato. Poiché il **regressore logistico** opera, alla fine della predizione, in uno spazio probabilistico, data l'azione della funzione sigmoide sul risultato logit della combinazione lineare, si è deciso, guardando alla letteratura

scientifica, di utilizzare la misura di **loss logaritmica** (Log-Loss in gergo tecnico). Si è utilizzata invece, una misura di **loss quadratica** per il training del singolo **regressore lineare**, in quanto, come è noto dalla sua modellazione matematica, esso opera in uno spazio "logit" non ristretto all'intervallo probabilistico [0,1]

- **Scelta del numero di batches del training set** : Il numero di batches scelto riflette la quantità di dati limitata di cui si costituisce il dataset. La scelta di suddividere il dataset in **5 mini-batch** è stata guidata dalla volontà di ottimizzare la frequenza di aggiornamento dei pesi senza sacrificare la stabilità del gradiente. Con una dimensione del batch di circa 31 campioni (considerando solo lo split di training), il modello **ottimizza** i parametri **5 volte per ogni epoca**, mantenendo, quindi, una varianza del gradiente bassa (guardando tutti i gradienti calcolati per ogni batch, all'interno della stessa epoca).

- **Scelta dei criteri di stop di training** : i criteri di stop sono stati scelti, per semplicità, in modo statico, come un **controllo su doppia condizione di tolleranza**, utilizzando due tolleranze di egual valore (e = 10^-1). In Particolare la discesa di gradiente si sarebbe fermata se : la norma del gradiente medio tra le batch dell'epoca corrente fosse stata minore di "e", oppure la norma della differenza tra i vettori dei parametri del regressore (prima e dopo l'epoca di ottimizzazione) fosse stata minore di "e". Nonostante il valore di tolleranza sia stato stabilito a priori, l'analisi a posteriori dei risultati mostra che il modello raggiunge una convergenza stabile in un numero congruo di epoche, confermando che la soglia di "e = 10^-1" è un iperparametro di training ragionevole per catturare i pattern significativi delle feature vocali senza eccedere nell'addestramento.
-

- **Scelta del "fixed" learning rate**: Il learning Rate utilizzato è stato **scelto a priori**, e impostato per semplicità a **n = 0.01**, senza effettuare controlli di validazione incrociata, affinché potessero essere esplorati i comportamenti dei regressori lineari e del modello ensemble con la modifica di altri iperparametri (vedi regolarizzazione e numero di boosting rounds). Il valore è stato impostato, in **maniera ragionevole**, per **bilanciare la velocità di discesa con la stabilità**: un valore troppo alto avrebbe causato oscillazioni attorno al minimo a causa della dimensione ridotta dei batch e della introduzione di rumore eccessivo nel gradiente (che comporterebbe cambi di direzione e possibili salti di minimo, se moltiplicato per un learning rate elevato), mentre un valore troppo basso avrebbe richiesto un numero eccessivo di epoche per la convergenza.

- **Scelta della regolarizzazione L2:** E' stato sperimentato l'utilizzo di una strategia di regolarizzazione ridge (L2), la quale permette (pesando con un iperparametro di regolarizzazione lambda i parametri quadrati del regressore) di penalizzare l'utilizzo di parametri di dimensione elevata, e, quindi, di mantenere una semplicità intrinseca del regressore (tramite riduzione della dimensione dei suoi parametri).

- **Scelta dell'iperparametro di regolarizzazione L2**: Una delle parti centrali dell'esperimento è stata, difatti, **l'esecuzione di una strategia di validazione incrociata** sul singolo regressore logistico e, successivamente, sul regressore lineare posto in sequenza con esso, per individuare l'iperparametro di regolarizzazione ridge più efficace per la generalizzazione e la stabilità per i due tipi di modello. Il parametro di regolarizzazione ridge per entrambi i modelli è stato scelto tramite una **validazione incrociata** su k**-folder di validazione** (k-folds cross-validation). Dato il training set derivato dal dataset normalizzato (Si fa notare che la validazione incrociata è stata eseguita su training set normalizzato derivante da dataset sbilanciato, e a parte, per confronto statistico, su training set derivante da dataset sotto-campionato), questo viene diviso in **n=6 splits (iperparametro da noi scelto),** e per mezzo dell'algoritmo **GroupKFold** messo a disposizione da scikit-learn, si itera k volte, addestrando il modello di regressione su k-1 folders, valutandolo sul k-esimo folder di validazione. In questo modo il modello viene validato sull'intero set di training, producendo ad ogni iterazione una misura della loss media dell'insieme di validazione, e poi mediando tutte le misure di loss derivanti da ogni iterazione.

  Le k iterazioni sono svolte per ogni iperparametro per cui si vuole validare il modello. nel nostro caso sono stati scelti **4 iperparametri di regolarizzazione ridge, lambdas = [0.001, 0.1, 1, 10]** ed è stato scelto per il modello di regressione logistica finale l'iperparametro **lambda = 0.001** che presentava la minore loss media tra i k folders di validazione e la minore deviazione standard (presentate in seguito le misure delle metriche, e la spiegazione dei risultati).

- **Scelta dell'iperparametro di regolarizzazione L2 (Regressore Lineare):** Seguendo la strategia di validazione incrociata dell'iperaparametro di regolarizzazione del regressore logistico, è stata eseguita la validazione incrociata sugli stessi parametri di regolarizzazione ridge, **lambdas = [0.001, 0.1, 1, 10]** sul regressore lineare. Tuttavia, in questo caso, la validazione incrociata dell'iperparametro è stata fatta fissando l'iperparametro di regolarizzazione del regressore logistico (weak learner precedente) al valore precedente, e addestrando il modello sui residui del predittore precedente (errori dati dalla classificazione probabilistica del regressore logistico), sempre seguendo la strategia di esecuzione delle k iterazioni con i k-1 folders di training e il k-esimo folder di validazione. Anche in questo caso, l'iperparametro con minore Mean Squared Loss tra i folders di validazione è stato **lambda = 0.001** (presentate in seguito le misure delle metriche, e la spiegazione dei risultati). **Un'altra strategia,** scartata per questioni di complessità di computazione (temporale)**,** sarebbe potuta essere l'esecuzione di una **validazione incrociata congiunta**, esplorando per coppie di iperparametri di regolarizzazione, le performance del modello additivo (primo weak learner logistico + secondo weak learner lineare).

- **Scelta dell'iperparametro di boosting (numero di rounds di boosting):** Un'altra decisione notevole nella configurazione del modello Ensemble è stata la scelta dell'iperparametro del numero di round di boosting. È stata effettuata una

validazione incrociata, nella quale sono stati validati su più folders diversi modelli Ensemble addestrati con il seguente numero di rounds di boosting **r = [2, 3, 5].** Dopo aver eseguito le validazioni incrociate (sui due dataset, bilanciato e non-bilanciato), il numero di rounds di boosting scelto è stato **n = 3.** Entrambi, difatti, mostravano una **curva di log-loss a "U" sul dominio di round r,** il quale portava, naturalmente, alla **scelta del parametro che minimizzava la metrica.** (Nella sezione di valutazione, si riporteranno i risultati, per diverse metriche, dei diversi rounds di boosting) .

- **Scelta di iperparametri non innnestata (in Cross Validation):** Una scelta di notevole interesse riguarda la semplificazione della **strategia di selezione degli iperparametri**. Invece di adottare una strategia di validazione incrociata innestata (nested cross-validation), nella quale si sarebbe esplorato lo spazio di **tutte le possibili terne di iperparametri** contemporaneamente, si è optato per una strategia sequenziale: prima è stato selezionato l'iperparametro di regolarizzazione lambda del regressore logistico, successivamente, fissato quest'ultimo al valore ottimale, è stato selezionato lambda del regressore lineare, e infine, fissati entrambi, è stato selezionato il numero di rounds di boosting. Questa scelta è stata motivata dalla volontà di ridurre il costo computazionale, che nella strategia innestata sarebbe cresciuto esponenzialmente con il numero di iperparametri da esplorare**. Il rischio teorico** di questa approssimazione è di **non individuare la combinazione globalmente ottimale degli iperparametri**; tuttavia, tale rischio è stato ritenuto accettabile data la dimensione limitata del dataset e la natura sequenziale dell'architettura Ensemble, nella quale ogni componente opera in uno spazio di predizione distinto (probabilistico per il regressore logistico, logit per i regressori lineari), rendendo ragionevole la separazione della loro ottimizzazione.

Valutazione

Le metriche adottate sono state:

- **mean log-loss (MLL)** : per valutare la misura di loss delle predizioni del regressore logistico (primo weak learner) e la misura di loss delle predizione del modello Ensemble.
- **Mean squared-loss (MSL) :** Per valutare la misura di loss delle predizioni dei modelli di regressione lineare intermedi, data la necessità di addestramento in uno spazio di predizione puramente logit.
- **Precision :** Per valutare il numero di veri positivi (target predetti correttamente e realmente tali) sul totale delle predizioni effettuate dal modello Ensemble, si fa notare che si è raccolta la misura di Precision per i true positives nella classe dei pazienti **con valore di target uguale a 1, quindi affetti dalla patologia**.
- **Recall :** Per valutare il numero di veri positivi sul totale dei target positivi presenti nel set di valutazione. si fa notare che si è raccolta la misura di Recall per i true positives nella **classe dei pazienti con valore di target uguale a 1**, quindi affetti dalla patologia.
- **F1 :** Metrica aggiuntiva, che valuta in maniera congiunta la precision e la recall

Di seguito si mostrano **le tabelle**, contenenti i risultati delle misure effettuate:

Tabella 1 — Cross-validation Regressore Logistico (Log-Loss)

Confronto della mean log-loss e deviazione standard per diversi valori di λ, su dataset bilanciato e non bilanciato.

λ	Mean Log-Loss (Non Bil.)	Std (Non Bil.)	Mean Log-Loss (Bil.)	Std (Bil.)
0.001	0.6682	0.0177	0.6397	0.0099
0.1	0.6682	0.0177	0.6397	0.0099
1.0	0.6682	0.0177	0.6398	0.0099
10.0	0.6684	0.0176	0.6407	0.0097

Tabella 2 — Cross-validation Regressore Lineare (Mean Squared Loss)

Confronto della mean squared loss e deviazione standard per diversi valori di λ, su dataset bilanciato e non bilanciato.

λ	Mean MSL (Non Bil.)	Std (Non Bil.)	Mean MSL (Bil.)	Std (Bil.)
0.001	0.1885	0.0489	0.1337	0.0186
0.1	0.1885	0.0489	0.1337	0.0186
1.0	0.1885	0.0488	0.1338	0.0185
10.0	0.1886	0.0486	0.1347	0.0185

Tabella 3 — Cross-validation Boosting Rounds (Dataset Non Bilanciato)

Metriche medie e deviazioni standard per diversi numeri di rounds di boosting sul dataset non bilanciato.

Rounds	Mean Log-Loss	Std	Mean Recall	Std Recall	Mean Prec.	Std Prec.	Mean F1	Std F1
2	0.3181	0.0196	0.9744	0.0628	0.6640	0.3258	0.7374	0.3262
3	0.3090	0.0275	0.9872	0.0444	0.6483	0.3065	0.7310	0.3071
5	0.3449	0.0506	0.9236	0.2386	0.5529	0.3187	0.6435	0.3231

Tabella 4 — Cross-validation Boosting Rounds (Dataset Bilanciato)

Metriche medie e deviazioni standard per diversi numeri di rounds di boosting sul dataset bilanciato.

Rounds	Mean Log-Loss	Std	Mean Recall	Std Recall	Mean Prec.	Std Prec.	Mean F1	Std F1
2	0.2906	0.0111	0.9306	0.1108	0.7419	0.2910	0.7790	0.2433
3	0.2728	0.0164	0.9395	0.0927	0.7419	0.2775	0.7845	0.2348
5	0.2857	0.0501	0.9041	0.2392	0.6217	0.3137	0.6921	0.2839

Tabella 5 — Risultati Finali su Test Set

Metriche di valutazione del modello Ensemble finale (r=3) sul test set, per entrambi i dataset.

Metrica	Dataset Non Bilanciato	Dataset Bilanciato
Log-Loss	0.3244	0.3183
Std Log-Loss	0.3513	0.3406
Recall	0.4839	0.7500
Precision	0.9375	0.4000
F1	0.6383	0.5217

Valutazione delle misure ottenute : La metrica primaria adottata per la selezione degli iperparametri è stata la Log-Loss (per il modello di regressione logistica e di Ensemble), in quanto metrica coerente con lo spazio di predizione probabilistica del modello: essa penalizza non solo la

classificazione errata, ma anche la scarsa assegnazione di probabilità alla classe target reale, risultando più informativa di metriche binarie come precision e recall nella fase di ottimizzazione.

Come mostrato nelle **Tabelle 1 e 2**, la cross-validation sull'iperparametro di regolarizzazione λ evidenzia differenze minime tra i valori esplorati, con una lieve tendenza alla crescita della loss per valori di λ elevati. Questo suggerisce che il modello non soffre di overfitting marcato, e che la regolarizzazione ha un ruolo di stabilizzazione piuttosto che di correzione. Il valore λ = 0.001 è stato selezionato per entrambi i regressori e su entrambi i dataset, in quanto minimizzava la mean loss tra i folder di validazione. Si fa notare, inoltre, che un valore così basso di regolarizzazione implica una penalizzazione quasi nulla del modello per quanto riguarda la dimensione dei suoi parametri e la complessità di quest'ultimo. Il dataset , infatti, ha solo 22 feature (dopo aver rimosso name e status), quindi lo spazio dei parametri è già limitato. Con pochi parametri e un dataset piccolo, **il rischio di overfitting** da parametri di grande dimensione è contenuto — il modello non ha abbastanza gradi di libertà per permettere un aumento di dimensione dei pesi.

Le **Tabelle 3 e 4** mostrano i risultati della cross-validation sul numero di rounds di boosting. In entrambi i dataset, r = 3 minimizza la log-loss media, con un **andamento crescente per r = 5** che indica l'insorgere di overfitting al crescere dei rounds. Le deviazioni standard elevate osservate su precision, recall e F1 sono attribuibili alla ridotta dimensione del dataset (195 istanze), che amplifica la varianza tra i folder di validazione.

La **Tabella 5** riporta i risultati del modello finale (r = 3) sul test set. Il confronto tra i due dataset rivela un aspetto interessante dell'esperimento: il modello addestrato sul dataset non bilanciato raggiunge una precision elevata (0.94) a scapito della recall (0.48), mentre il modello addestrato sul dataset bilanciato inverte questo rapporto (recall 0.75, precision 0.40). Questo trade-off, emerso naturalmente dal bilanciamento del dataset tramite undersampling, conferma l'importanza della fase di preprocessing nella definizione del comportamento del modello, e motiva l'integrazione con il modulo di validazione della KB, progettato proprio per rilevare e segnalare i casi di falso negativo critico.

Le **log-loss medie del regressore logistico**, inoltre, confrontate con le l**og-loss medie del modello Ensemble**, dimostrano la capacità del modello Ensemble di scorgere ulteriori pattern, dopo l'addestramento dei weak learners sui residui predittivi del primo regressore logistico. Infatti, le log loss risultanti dalla fase di validazione incrociata del primo modello sono alte, indicative di una capacità predittiva limitata dalla linearità del modello rispetto alla complessità del problema. Le misure di loss tendono a stabilizzarsi ad un risultato notevolmente minore nelle predizioni del modello Ensemble, in quanto capace, per mezzo di addestramento dei weak learners successivi (regressori lineari intermedi) sui residui del primo regressore logistico, di correggere la direzione della predizione probabilistica del modello finale.

Rappresentazione della conoscenza

Sommario

Il secondo modulo del sistema è una Knowledge Base implementata in **linguaggio Prolog**, integrata nell'ambiente Python tramite la libreria **pyswip**. La KB non è utilizzata come semplice database di fatti, bensì come sistema inferenziale capace di ragionare sul profilo vocale del paziente, derivare sintomi impliciti attraverso una gerarchia semantica, calcolare la severità della diagnosi e validare o confutare la predizione probabilistica del modello Ensemble. La conoscenza di dominio è modellata su evidenze cliniche consolidate riguardanti valori di soglia di specifici marker vocali, quali **jitter, shimmer, hnr, nhr**.

Jitter (Perturbazione di Frequenza): Valuta la variazione di tono da un ciclo all'altro. Un jitter elevato indica una voce rauca o instabile.

Shimmer (Perturbazione di Ampiezza): Valuta la variazione di volume da un ciclo all'altro. Un shimmer elevato è associato a ruvidità vocale o respiro affannoso.

HNR (Rapporto Armoniche/Rumore): Misura il rapporto tra componenti periodiche (armoniche) e componenti non periodiche (rumore) nella voce.

NHR (Rapporto Rumore/Armoniche): L'inverso di HNR, che misura il rapporto tra rumore e componenti armoniche. Sono preferiti valori più bassi.

La KB viene quindi modellata per mezzo di clausole di Horn, che permettono di esprimere sia fatti deterministici che regole di inferenza complessa.

Strumenti utilizzati

Il modulo della Knowledge Base è stato realizzato per mezzo del linguaggio di programmazione logica Prolog. Esso consente di istanziare fatti e costruire regole di inferenza complessa, utilizzando **la logica del primo ordine** e le **clausole di Horn**.

SWI-Prolog e linguaggio Prolog. Il linguaggio Prolog è stato scelto come linguaggio di implementazione della KB per la sua natura dichiarativa e per la disponibilità di meccanismi di risoluzione delle query, quali **unificazione, backtracking e inferenza basata su regole**, caratteristiche che lo rendono adatto alla modellazione di conoscenza medica strutturata. La KB è stata

implementata tramite clausole di Horn, che costituiscono il sottoinsieme della logica del primo ordine su cui si basa il linguaggio Prolog [6, Cap. 15].

pyswip. L'integrazione tra la KB Prolog e il modulo Python è stata realizzata tramite la libreria pyswip, che espone un'interfaccia Python per la consultazione di una KB SWI-Prolog [5] e l'esecuzione di query dall'ambiente Python. E' stato scelto l'utilizzo di pyswip per la **necessità di integrare le query sulla base di conoscenza** (scritta in Prolog) con il resto del sistema (Modello Ensemble e classificazione probabilistica), interamente implementato in python. Tramite pyswip è possibile istanziare un oggetto Prolog, caricare la KB tramite il metodo consult, asserire fatti temporanei con assertz, eseguire query e ritirare i fatti con retract, permettendo quindi di valutare la situazione clinica di ogni paziente, asserire i fatti clinici espliciti e il risultato di predizione del modello di apprendimento automatico precedente (Ensemble), e inferire il risultato e il report, tramite regole di inferenza complesse.

Decisioni di Progetto

La KB è organizzata in due blocchi distinti: **fatti e regole**. I fatti codificano la conoscenza statica del dominio clinico, mentre le regole permettono di inferire risultati astratti riguardanti la presenza di feature specifiche (a partire dai valori espliciti delle features di ogni paziente), e determinare la severità della diagnosi, l'inconsistenza dei valori delle features del paziente, e la possibilità di diagnosi non affidabile.

Di seguito si elencano le diverse scelte di progetto e la descrizione della KB:

- **Scelta delle feature vocali e delle soglie cliniche:** Le feature selezionate per la KB sono **Jitter:DDP, Shimmer:APQ3, NHR e HNR**, estratte dal dataset originale. La scelta è motivata dalla loro rilevanza clinica documentata nella letteratura sul Parkinson [1, 2, 4]: il Jitter:DDP misura la perturbazione ciclo-per-ciclo della frequenza fondamentale del segnale vocale [1], lo Shimmer:APQ3 misura la perturbazione dell'ampiezza su tre cicli consecutivi, il NHR quantifica il rapporto rumore/armonici e l'HNR il suo inverso. Le soglie utilizzate per classificare i valori come anomali — **Jitter:DDP > 0.0057, Shimmer:APQ3 > 0.030, NHR > 0.015, HNR < 20.0** — sono state derivate dalla letteratura clinica [2,3], e non dal dataset stesso, permettendo alla base di conoscenza di risultare indipendidente dalla specifica struttura del dataset, e di modellare la KB con conoscenza di fondo, utile per validare o smentire le predizioni del modello Ensemble, addestrato unicamente sui dati e quindi suscettibile a bias della struttura di dati selezionata.

- **Scelta della rappresentazione simbolica delle feature:** Le regole **has_feature(..)** traducono i valori numerici espliciti delle feature, importate direttamente dal dataset non normalizzato, in simboli logici (**high_jitter, high_shimmer, high_nhr, low_hnr**) applicando le soglie cliniche. Questa separazione tra livello numerico e livello simbolico è una scelta progettuale importante: tutte le regole di ragionamento di livello superiore operano esclusivamente su simboli, mantenendo la KB indipendente dai valori specifici del dataset, evitando di ragionare esclusivamente sui valori per eseguire del mero pattern matching (col fine ultimo e banale di determinare la condizione del paziente).

- **Scelta della gerarchia semantica dei sintomi:** I sintomi sono organizzati in una gerarchia semantica tramite i fatti **is_subtype_of(..)**: vocal_tremor e breathy_voice sono sottotipi di dysphonia, vocal_tremor è anche sottotipo di vocal_instability, e vocal_instability è a sua volta sottotipo di dysphonia. Questa struttura non è utilizzata come semplice gerarchia statica, ma come base per la propagazione dell'inferenza di un sintomo più "astratto" tramite la regola **has_symptom_or_super(..).** Questa regola di inferenza permette alla KB di comprendere se la prima feature individuata a partire dai valori istanziati dei bio-marcatori vocali è legata, lungo la gerarchia, a uno specifico sintomo della patologia analizzata.

- **Scelta dei pesi clinici dei sintomi:** I sintomi non sono trattati come equivalenti nella valutazione della severità diagnostica: **vocal_tremor** riceve **peso 3**, **hoarse_voice** e **breathy_voice peso 2**. Questa scelta riflette l'evidenza clinica per cui il tremore vocale è una feature di alto livello più discriminante [4] rispetto alla raucedine o alla voce soffocata, che possono essere causati anche da altre patologie. Il punteggio pesato prodotto da **weighted_diagnosis(..)** deSWItermina il livello di severità della diagnosi in quattro categorie (severe, moderate, mild, none), con soglie impostate a :

    - **score >= 5 per severe**.
    - **>= 3, <5 per moderate.**
    - **>= 1, <3 per mild**.

- **Scelta del criterio di affidabilità della diagnosi:** La regola **reliable_diagnosis(..)** introduce un **controllo di affidabilità indipendente dal punteggio pesato**: una diagnosi è considerata affidabile solo se supportata da almeno due sintomi distinti. Questa scelta è dettata dalla necessità di prevenire che sia eseguita una diagnosi, soltanto dopo che la KB ha inferito una sola feature di alto livello relata alla patologia del parkinson. Il predicato **unreliable_diagnosis(..),** infatti, rileva il caso appena descritto: se lo score non pesato (conteggio) delle features di alto livello **si riduce ad una sola feature**, allora questo determina una diagnosi da parte della KB non affidabile per valutare la coerenza della sua diagnosi con la predizione del modello ML.

- **Scelta dei pattern di inconsistenza clinica:** Sono stati modellati tre regole di inferenza di **inconsistency nei dati** tramite **NAF (Negation As Failure**): **jitter isolato** (**high_jitter** senza high_shimmer né high_nhr), **HNR basso isolato** (**low_hnr** senza altri marcatori anomali) e **NHR elevato isolato** (**high_nhr** senza altri marcatori anomali). Questi pattern corrispondono a situazioni in cui un singolo marcatore anomalo non è supportato da evidenze attese, invece, in un profilo parkinsoniano), rendendo il profilo vocale clinicamente contraddittorio rispetto al pattern atteso del Parkinson. In questo modo si hanno due tipi di regole di inferenza diverse : una per l'inferenza di una diagnosi non affidabile (sopra descritta), dedotta da mancate evidenze di sintomi e valori particolarmente sopra/sotto soglia, e una per l'inferenza di **un risultato** di un

avviso inferito dalla KB, riguardo la **presenza di dati conflittuali**, ovvero pattern anomali nella disposizione dei valori di features nel paziente.

- **Scelta di inferirenza di tipi vocali :** È stato introdotto il predicato **vocal_profile(..),** che classifica il paziente in uno di quattro profili vocali : **tremor_dominant, hoarseness_dominant, mixed_dysphonia, breath_dominant** attraverso combinazioni di feature con uso esplicito della **NAF (Negation As Failure) (\+)**. Ad esempio, tremor_dominant richiede la co-presenza di high_jitter e high_shimmer in assenza di low_hnr, distinguendolo da mixed_dysphonia che richiede invece la co-presenza di high_jitter e low_hnr. La scelta di modellare esplicitamente i profili vocali risponde alla necessità precisa di separare i casi di predizione falsamente negativa da parte del modello di apprendimento automatico. Infatti, una classificazione (0) da parte del modello di predizione precedente, data l'evidenza di un profilo clinico **tremor_dominant,** e una diagnosi severa (quindi con più features di più alto livello inferite), inferisce un **errore di severità critica** da parte del modello di predizione.

- **Scelta dei predicati di rilevazione e classificazione degli errori critici:** I predicati **critical_false_negative(..)** e **critical_false_positive(..)** identificano i casi di errore critico del modello ML confrontando la predizione con la diagnosi deduttiva della KB. Un falso negativo critico si verifica quando la KB produce una diagnosi severe ma il modello predice 0 (paziente sano): un caso pericoloso, in quanto un paziente con forte evidenza di Parkinson (dalle sole features vocali) viene classificato erroneamente come sano. Un falso positivo critico si verifica invece quando la KB produce una diagnosi none ma il modello predice 1: il modello segnala Parkinson in assenza di qualsiasi evidenza acustica rilevabile dalla KB. La severità dell'errore critico viene ulteriormente divisa in categorie dal predicato **critical_error_severity(..),** già precedentemente descritto, che combina il tipo di errore con il profilo vocale del paziente.

- **Scelta del sistema di validazione a priorità decrescente.** Il predicato **validation(..)** integra tutti i moduli precedenti in un sistema di validazione che assegna un risultato alla predizione del modello ML, confrontandola con la diagnosi del sistema di inferenza della KB, secondo una gerarchia di priorità decrescente, implementata tramite if-then-else (messo a disposizione dal linguaggio): **errore critico severo > errore critico moderato > errore critico generico > evidenza inaffidabile > dati conflittuali > predizione coerente > warning**. La scelta di una gerarchia esplicita garantisce che la discordanza più critica tra la predizione del modello Ensemble e la diagnosi effettuata dalla KB venga sempre segnalata con priorità, evitando che un errore intermedio di classificazione del modello sia segnalato prima di un errore critico severo (per esempio). In particolare, i falsi negativi su pazienti con profilo **tremor_dominant** ricevono la priorità massima, in quanto rappresentano il rischio clinico più grave: un paziente con una feature di più alto livello strettamente legata alla patologia, ma classificato come sano dal modello di classificazione probabilistica.

Valutazione

Si rimanda la valutazione dei risultati ottenuti con l'integrazione della base di conoscenza e del ragionamento automatico alla successiva sezione di argomento (Ragionamento automatico). Per questa sezione, invece ci si concentra sulla struttura e sulla relativa complessità della base di conoscenza.

La **complessità strutturale** della KB può essere valutata analizzando la composizione dei suoi elementi. La base di fatti è composta da *13 fatti espliciti*, suddivisi in quattro categorie: 1 associazione diretta sintomo-malattia, *4 mapping feature-sintomo* tramite **indicates(..),** 3 pesi clinici dei sintomi tramite **symptom_weight(..),** e *5 relazioni gerarchiche* tramite **is_subtype_of/(..).** La base di regole comprende 24 predicati distinti, organizzati su più livelli di astrazione crescente: dal livello più basso di classificazione numerica delle feature **(has_feature(..)),** fino al livello più alto di validazione e spiegazione tramite evidenza delle features di alto livello individuate dalla KB **(validation(..), weighted_validation(..), report(..), explain(..)).**

La gerarchia semantica dei sintomi si sviluppa su tre livelli di profondità nel caso di **vocal_tremor** : **vocal_tremor → vocal_instability → dysphonia**

La gerarchia semantica si sviluppa, invece, su due livelli nel caso di **breathy_voice** e **hoarse_voice.** Questa struttura, combinata con la regola ricorsiva **has_symptom_or_super(..),** permette al sistema di propagare l'inferenza di un sintomo di basso livello lungo la gerarchia.

La KB modella **4 profili vocali distinti, 4 livelli di severità diagnostica**, **3 pattern di inconsistenza** clinica e **3 livelli di errore critico**. Il risultato complessivo della validazione può assumere 7 valori distinti : **severe_critical_error, moderate_critical_error, critical_error, unreliable_evidence, conflictual_data, coherent, warning.**

Ciascuno dei risultati possibili dalla diagnosi della KB possiede una semantica precisa. Questo permette al sistema di non determinare, banalmente, un report del tipo coerente/incoerente in base alla predizione del modello ML, ma di giustificare con più livelli di spiegazione, la diagnosi effettuata e il confronto con la predizione precedentemente effettuata in maniera probabilistica.

Ragionamento automatico

Sommario

Il terzo modulo del sistema è il componente di ragionamento automatico, responsabile dell'integrazione tra il modello Ensemble e la Base di Conoscenza. Il modulo orchestra il flusso di esecuzione paziente per paziente: ottiene la predizione probabilistica del modello ML, istanzia i valori vocali del paziente come fatti temporanei nella KB, esegue le query di validazione tramite il motore inferenziale Prolog, e classifica il risultato della validazione. Il ragionamento si avvale dei meccanismi di unificazione, backtracking e risoluzione SLD per derivare il risultato di validazione risalendo la catena inferenziale definita nella KB. La realizzazione del modulo ha attraversato le seguenti fasi principali:

- Implementazione dell'interfaccia **Python-Prolog** tramite pyswip per la gestione del ciclo assertz/query/retract.
- Definizione del flusso di integrazione tra predizione del modello Ensemble e interrogazione della KB.
- Implementazione del predicato **report(..)** per la produzione di output esplicabile.
- Valutazione del sistema ibrido sul test set tramite analisi delle proporzioni delle categorie di risultato.

Strumenti utilizzati

l modulo di ragionamento automatico è implementato in Python tramite la classe **KBInterface**, che incapsula l'interfaccia con la **KB Prolog**. L'integrazione è realizzata tramite la **libreria pyswip** [5], già descritta nella sezione precedente. Il modulo utilizza inoltre le stesse strutture dati del modulo di apprendimento automatico, ovvero **dataframe pandas** per la gestione delle righe del dataset. Non sono stati utilizzati algoritmi di ragionamento originali: il ragionamento è interamente delegato al motore inferenziale della KB Prolog, che esegue **risoluzione SLD sulle clausole di Horn della KB** [6, Cap. 15]

Decisioni di Progetto

Scelta dei metodi assertz/retract per l'istanziazione dei dati del paziente: Per ogni paziente del test set, i valori delle quattro feature vocali scelte per la diagnosi nella KB, vengono istanziati temporaneamente come fatti nella KB tramite **assertz** (metodo messo a disposizione dalla libreria pyswip), consentendo alle regole Prolog di operare su di essi attraverso i predicati **jitter_value(..), shimmer_value(..), hnr_value(..) e nhr_value(..).** Al termine della query, i fatti vengono rimossi tramite **retract**, ripristinando lo stato originale della KB. Questo permette alla KB di non istanziare fatti relativi a pazienti precedentemente analizzati, rimanendo in stato “clean” dopo ogni interrogazione sulla base di conoscenza.

Scelta di separare la query result_only dalla query report completo: Il metodo query_result_kb espone due modalità di interrogazione controllate dal parametro result_only. Nella modalità **result_only=True**, viene eseguita esclusivamente la query **validation(..),** restituendo solo il risultato della validazione. Nella modalità **result_only=False**, viene eseguita la query **report(..),** che produce

un output semanticamente ricco, contenente: **la forza dell'evidenza** (kb_evidence_strength), **il risultato della validazione** (critical_error, conflictual_data, etc…), **le evidenze pesate** (sintomi rilevati con i relativi pesi) e **la predizione del modello ML**. In questo modo, nei contesti in cui è necessario solo il risultato della validazione, il validatore può decidere di interrogare la KB utilizzando una query con risultato di validazione non esteso e più diretto.

Scelta del predicato report(..) per la spiegabilità dell'output: Il predicato report(..) è stato progettato per produrre un risultato di validazione, confutazione interpretabile, che non si limita a restituire il risultato della validazione ma fornisce anche la motivazione del risultato: le evidenze di features acustiche rilevate con i rispettivi pesi, la forza complessiva dell'evidenza e la predizione del modello ML. Questa scelta è motivata dal contesto clinico del sistema: un KBS per la diagnosi assistita deve essere non solo predittivo, ma deve garantire anche **la spiegabilità del suo risultato**, permettendo all'esperto, che consulta il sistema, di comprendere su quali basi il sistema ha prodotto un determinato risultato. Il messaggio prodotto da report(..) integra quindi le informazioni provenienti da :

- **weighted_validation(..),** che combina il risultato della validazione con la forza dell'evidenza
- **explain(..),** che raccoglie tramite **findall(..)** (metodo built-in per iterare e raccogliere in una lista dei valori inferiti dalla KB) i sintomi rilevati con i loro pesi clinici.

Scelta di utilizzare i valori non normalizzati per la KB: Il modello Ensemble opera sui valori normalizzati delle features (**Z-score normalization**), mentre la KB opera sui valori originali, prelevati direttamente dal dataset originale. Questa scelta è obbligata: le soglie cliniche derivate dalla letteratura (Jitter:DDP > 0.0057, ecc.) sono definite nello spazio dei valori reali, non in quello normalizzato. Il modulo di ragionamento gestisce quindi due dataframe paralleli per ogni paziente : il dataframe normalizzato per la predizione ML e il dataframe non normalizzato per la query KB. Seleziona, quindi, le feature vocali rilevanti tramite il metodo **select_features_from_dataframe_row(..)** prima di procedere all'**assertz**.

Scelta del flusso di integrazione ML + KB: Il modulo run_kb_validator.py permette il flusso completo di integrazione: per ogni riga del test set, **il modello Ensemble,** addestrato sul training set di riferimento, produce una predizione probabilistica continua, che viene **discretizzata** tramite la funzione **binary_classify**(..), funzione statica del regressore logistico, che **tramite un threshold**, trasforma la predizione continua in 0 o 1; la predizione discreta viene quindi passata alla KB insieme ai valori vocali del paziente per la validazione. Questo flusso è replicato separatamente per il dataset bilanciato e non bilanciato, permettendo il confronto delle proporzioni di validazione tra i due contesti di addestramento. I risultati vengono aggregati contando le occorrenze di ciascuna categoria di risultato e calcolando le proporzioni sul totale del test set.

Valutazione

In questa sezione di valutazione si mostrano i risultati dell'integrazione del modello di KB con il modello Ensemble iniziale. Per entrambi i dataset (bilanciato e non bilanciato) sono state generate delle predizioni per ogni sample di paziente, e poi ogni predizione, con le relative features di input (features vocali dei pazienti), è stata validata o confutata dalla KB, corredandola da evidenza di feature individuate, con relativo peso associato all'interno della diagnosi, tipo di errore critico di

predizione (se individuato), analisi delle conflittualità tra valori delle features selezionate per la diagnosi (tipo di conflittualità individuata).

I risultati della validazione della KB sono riportati nella **Tabella 6**, che mostra le proporzioni delle categorie di risultato sul test set per entrambi i dataset.

Tabella 6 — Risultati validazione KB sul test set

Proporzioni delle categorie di risultato della validazione KB per dataset bilanciato e non bilanciato.

Categoria	Dataset Non Bilanciato	Dataset Bilanciato
Coherent	**65.1% (28/43)**	**53.8% (14/26)**
Conflictual data	**30.2% (13/43)**	**11.5% (3/26)**
Critical error	**2.3% (1/43)**	**30.8% (8/26)**
Unreliable evidence	**2.3% (1/43)**	**3.8% (1/26)**
Warning	**0.0%**	**0.0%**

La proporzione di predizioni coerenti è elevata in entrambi i datasets, 65.1% sul non bilanciato e 53.8% sul bilanciato, indicando che nella maggioranza dei casi il ragionamento deduttivo della KB è allineato con la predizione probabilistica del modello Ensemble.

La differenza più significativa tra i due dataset riguarda la proporzione di errori critici: 2.3% sul dataset non bilanciato contro 30.8% sul dataset bilanciato. Questo risultato è direttamente coerente con le misure di recall/precision (calcolate per la classe di pazienti con valore target positivo) osservato nella valutazione del modello ML.

Guardando i risultati di validazione della KB e le percentuali mostrate precedentemente, il modello addestrato sul dataset non bilanciato tende a predire la classe negativa (paziente sano) con alta precisione ma bassa recall, producendo molti falsi negativi che tuttavia, in questo caso, non vengono rilevati come critici dalla KB perché i pazienti corrispondenti non presentano evidenza acustica forte. Il modello addestrato sul dataset bilanciato, invece, ha recall più alta ma precision più bassa — predice positivo più spesso — e questo si riflette in una proporzione di errori critici molto più alta, corrispondenti a falsi positivi rilevati dalla KB come predizioni di Parkinson in assenza di evidenza acustica (diagnosis none o mild con predizione 1).

La proporzione elevata di dati conflittuali sul dataset non bilanciato (30.2%) indica che molti pazienti del test set presentano un singolo marcatore anomalo isolato, senza conferme complementari. Questo suggerisce che il dataset contiene una quota significativa di profili vocali ambigui, non chiaramente riconducibili al pattern acustico tipico del Parkinson, e che la KB riesce a identificarli e segnalarli correttamente.

Conclusioni

Il sistema sviluppato integra un modello di apprendimento automatico, un Gradient Boosting Ensemble , con una Base di Conoscenza in Prolog per la diagnosi assistita della malattia di Parkinson a partire da features vocali. I risultati ottenuti mostrano che il sistema ibrido è in grado di produrre predizioni probabilistiche e di validarle attraverso ragionamento deduttivo, fornendo un output non solo predittivo ma anche clinicamente interpretabile.

Tra le principali limitazioni del sistema si segnalano: la **ridotta dimensione del dataset (195 istanze**), che produce deviazioni standard elevate nelle metriche di cross-validation e limita la significatività statistica dei risultati; la KB opera su sole quattro feature vocali, mentre il dataset originale ne contiene 22 — **un'estensione della KB a un insieme più ampio di marcatori potrebbe migliorare la copertura del ragionamento deduttivo**; le soglie cliniche sono fisse e non adattive, mentre una strategia di apprendimento delle soglie direttamente dai dati potrebbe aumentare la sensibilità del sistema.

Sviluppi futuri potrebbero includere l'integrazione di tecniche di spiegabilità più avanzate per il modello ML, l'estensione della KB a una ontologia clinica più completa, e la validazione del sistema su dataset più ampi e provenienti da contesti clinici reali.

Sul fronte della KB, la valutazione ha mostrato che la proporzione di predizioni coerenti è elevata in entrambi i dataset (65.1% e 53.8%), confermando che il ragionamento deduttivo è nella maggioranza dei casi allineato con la predizione probabilistica. La differenza nella proporzione di errori critici, 2.3% sul non bilanciato contro 30.8% sul bilanciato, riflette direttamente il trade-off recall/precision del modello ML, dimostrando la coerenza tra i due moduli del sistema. La proporzione elevata di dati conflittuali sul dataset non bilanciato (30.2%) indica che il dataset contiene una quota significativa di profili vocali ambigui, che la KB riesce a identificare e segnalare correttamente.

Sul fronte del modello ML, la cross-validation ha evidenziato che il numero ottimale di rounds di boosting è r=3 per entrambi i dataset, con una **log-loss media** di **0.309** sul dataset non bilanciato e 0.273 sul bilanciato. Il confronto tra i due dataset ha rivelato un trade-off clinicamente rilevante: il modello addestrato sul dataset non bilanciato raggiunge **precision elevata** (0.94) a scapito della **recall** (0.48), mentre il modello bilanciato inverte questo rapporto (recall 0.75, precision 0.40). In un contesto di diagnosi precoce, una recall bassa implica un elevato numero di falsi negativi — pazienti malati classificati come sani — il che rappresenta un rischio clinico superiore. **Questo trade-off motiva l'integrazione con la KB**, progettata proprio per rilevare e segnalare i casi critici che il modello ML non riesce a classificare correttamente.

Riferimenti Bibliografici

[1] Little, M. A., McSharry, P. E., Hunter, E. J., Ramig, L. O. (2008). Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Transactions on Biomedical Engineering, 56(4), 1015-1022.

[2] Viswanathan, R., et al. (2022). Phonemes based detection of Parkinson's disease for telehealth applications. Scientific Reports, 12, 9591. https://doi.org/10.1038/s41598-022-13865-z

[3] Mathew, R., et al. (2025). Voice biomarkers as prognostic indicators for Parkinson's disease using machine learning techniques. Scientific Reports, 15, 11385. https://doi.org/10.1038/s41598-025-96950-3

[4] Quan, C., et al. (2020). The physical significance of acoustic parameters and its clinical significance of dysarthria in Parkinson's disease. Scientific Reports, 10, 11396. https://doi.org/10.1038/s41598-020-68754-0

[5] Leuenberger, Y. (2008). pyswip — Python SWI-Prolog bridge. https://github.com/yuce/pyswip

[6] Russell, S., Norvig, P. (2021). Artificial Intelligence: Foundations of Computational Agents (4th ed.). Pearson. [Cap. 7 — Apprendimento Supervisionato; Cap. 15 — Rappresentazione con Logica del Primo Ordine]