Università Politecnica delle Marche

Corso di Laurea Magistrale in **Ingegneria Elettronica**

Progetto per il corso
Wireless Sensor Networks for IoT

Analisi del tremore

Data:

16/12/2024

Autori:

Raul Fratini, Alessia Conti, Fulvio Michele Luigi Buono,

Alessandro Polverari, Alex Voltattorni, Lorenzo Mozzoni

Indice

Introduzione

Abstract

Dato il rapido espandersi dell'uso in ambito sanitario di dispositivi wearable, predisposti al monitoraggio dei sintomi motori nella malattia di Parkinson, la letteratura scientifica si pone l'obiettivo di effettuare in primo luogo una classificazione binaria per riconoscere lo stato di tremore o non tremore del soggetto e poi, una volta individuato il tremore, fornire una correlazione con la scala UPDRS. La sigla UPDRS sta per 'Unified Parkinson's Disease Rating Scale' ed è una scala utilizzata nella valutazione della prognosi della malattia del Parkinson; questa è costituita da valori compresi tra '0' e '4', in cui il livello '0' è associato alla mancanza di tremore fino al livello '4', che corrisponde alla situazione più grave.

Verranno dapprima analizzate nello stato dell'arte le metodologie di analisi proposte in letteratura, per inquadrare il contesto generale e successivamente sarà affrontato l'aspetto sperimentale con l'obiettivo di confrontare i risultati ottenuti con un classificatore basato su soglie con quelli del Classification Learner di Matlab.

I dati raccolti sono stati analizzati ed elaborati in Matlab attraverso operazioni di pre-processing, estrazione di feature e sviluppo di un classificatore; infine, i risultati sono stati confrontati con quelli trovati in letteratura per trarre le relative conclusioni.

Stato dell'arte

Tramite l'utilizzo di reti di sensori, che sono in grado di quantificare i sintomi motori sia negli arti inferiori che in quelli superiori, ma anche nel parlato, e sfruttando unità inerziali, come accelerometro e giroscopio, è possibile ottenere algoritmi di classificazione del tremore.

Mentre per gli arti inferiori viene effettuata un'analisi del cammino e dell'attività motoria, per gli arti superiori, che costituiscono il tema principale dell'elaborato, viene proposta una valutazione quantitativa del tremore in base alla posizione del dispositivo. Perciò, etichettando con algoritmi di intelligenza artificiale le finestre che compongono l'acquisizione, è possibile stabilire se la camminata sia regolare e sciolta oppure affetta da sintomi motori (per esempio un congelamento del passo). Il fine indiretto è tentare di personalizzare la terapia del Parkinson relativamente alla gravità dei sintomi dei pazienti e allo stato di avanzamento della malattia attraverso un'analisi il più possibile oggettiva e puntuale. Per fare questo si acquisiscono dei dati attraverso l'uso di un accelerometro e dopo averli processati, vengono associati al corrispondente valore della scala UPDRS.

I test per la valutazione del tremore che il neurologo sottopone al paziente possono essere di tre tipologie: i test a riposo sono quelli in cui il soggetto tiene le braccia appoggiate sulle gambe; nei test posturali deve tenere le braccia distese davanti a sé; nei test cinetici si valuta la capacità del paziente di toccarsi la punta del naso con l'indice, in particolare se il movimento avvenga in modo regolare o meno.

In letteratura sono già stati implementati algoritmi di intelligenza artificiale che superino il limite imposto da classificatori basati su soglia fissa per ottimizzare il processo complessivo.

Work	Year	Sensors and Location	Participants	Duration	Analysis Methods	Main Aims or Findings	Main Results
Adams et al. [1 - dataset di riferimento]	2021	5 sensors, one on each limb and on the trunk	17 PD, 17HS	3s	Cross-correlation analysis	Analyze tremor characteristics of participants inside and outside the clinic	Tremor was present for 1.6 hours per day in most affected hands of PD patients
Sigcha et al. [2]	2021	Custom-built mHealth mobile and wearable application for tracking motor symptoms of PD patients using smartphones and smartwatches	18 PD patients	2.56 s window (128 samples) with a 50% overlap	CNN model	The use of consumer smartwatches and their embedded accelerometers for monitoring and evaluating resting tremor in PD patients	LOSO evaluation show a sensitivity and specificity of 86.1%
Zhao et al. [3]	2024	The wearable sensor shimmer3 IMU units	85 PD subjects	1.5 s window	LGBM, SVM, KNN, XGB	A novel technical pipeline is proposed for fine-grained classification of PD severity grades	The fine-grained classification accuracy 82.41%
Crowe et al. [4]	2024	2 IMUs on the most affected wrist and ankle	24 PD	3s, 6s, 12s	Supervised ML	Demonstrate the feasibility of detecting symptom events in free-living setting	Accuracy of 83%, 75%, 81% for tremor, bradychinesia and discynesia
San-Segundo et al. [5]	2020	Wrist-worn triaxial accelerometers	12 patients with PD	3-second windows (150 samples per window) with a 2-second overlap	Deep learning: Convolutional neural networks	Evaluation of novel preprocessing methods and algorithms in free-living and laboratory settings	Error lower than 5% when estimating the percentage of tremor in a laboratory setting

Work	Year	Sensors and Location	Participants	Duration	Analysis Methods	Main Aims or Findings	Main Results
Rigas et al. [6]	2012	6 accelerometers located on the wrists, ankles, sternum, and the waist	23 subjects (18 PD patients and 5 healthy controls)	3 s duration window and 1.5 s overlapping	Hidden Markov model	High accuracy in tremor detection. It is possible to discriminate tremors from other PD symptoms	Accuracy of 87% for tremor severity, maximum specificity 97%, and sensitivity 95%
Sun et al. [7]	2021	Ultigesture sensors and developed the TremorSense Android application	30 PD patients	Sensor data with a window size of 1.28 seconds and a sliding window size of 0.64 seconds	8-Layer CNN model	Some wearable devices demand accurate tremor detection to provide appropriate mitigations and treatments	CNN model with self-evaluation, cross-evaluation and leave-one-out evaluation, and the accuracies for all three evaluations are greater than 94%
Smid et al. [8]	2022	2 wired tri-axial accelerometers attached to the index fingers	54 subjects (28 PD patients and 26 controls)	X	Threshold method using a statistical analysis	MDS-UPDRS tremor tests can be translated to objective accelerometric measurements	69.6% concordance with MDS-UPDRS ratings
Duque et al. [9]	2020	Gyroscope place on 2 different arm positions in rest and posture position	19 PD, 20 ET, 12 HS	3s	ML algorithms	Improve differential diagnosis between patients with PD and ET	Accuracy of $97.2 \pm 3.7\%$ to differentiate between healty and trembling subjects

Work	Year	Sensors and Location	Participants	Duration	Analysis Methods	Main Aims or Findings	Main Results
Bremm et al. [10]	2024	2 inertial measurements units on the dorsal part of each hand	33 PD, 12 controls	5s	Supervised ML	Classify and predict MDS-UPDRS scores for 6 MDS-UPDRS tasks	Accuracy of 94%
Hssayeni et al. [11]	2019	2 motion sensor (tri-axial accelerometer and gyroscope) mounted on the wrist and on the ankle	24 PD	5s	Gradient tree boosting and DL	Estimate Parkinsonian tremor as the patients performed a variety of free body movements	$r = 0.96$ for GTB and $r = 0.84$ for DL

Materiali e metodi

Dataset

Il dataset utilizzato per sviluppare il progetto, reperibile online ed etichettato dalla supervisione dei neurologi, è denominato “PD-BioStamp RC 21” [1] e contiene dati derivanti da un sensore accelerometrico utilizzati per studiare il tremore e altri sintomi motori presenti sia in soggetti con malattia di Parkinson che in soggetti cosiddetti ‘di controllo’.

Il dataset è formato da più documenti, in particolare vi è un file denominato ‘Clinic_Data_PD_BioStampRCStudy.csv’ che contiene tutte le informazioni relative ai soggetti considerati e include: l’identificativo di ogni singolo soggetto che costituisce il dataset, il sesso, lo status (cioè se si tratta di un soggetto parkinsoniano oppure di controllo), l’età, la tipologia di test cui è stato sottoposto (per esempio l’etichetta ‘3_17’ indica che si tratta di un test a riposo, la lettera definisce la posizione del sensore e la dicitura on/off si riferisce alla medicazione). Nella tabella viene riportato il valore UPDRS associato a ciascuna prova svolta, per ogni paziente.

Gli altri file, sempre in formato ‘.csv’, contengono tutti i dati, divisi per paziente, sulle prove che ciascuno di loro ha svolto; in particolare viene specificato il tipo di test, le condizioni in cui è stato fatto e gli istanti di inizio e di fine. In aggiunta, per ogni paziente sono presenti dei file, ciascuno per ogni parte del corpo in cui era stato posizionato il sensore (petto, arti inferiori e superiori), che contengono tutti gli istanti temporali relativi ad ogni misurazione e i corrispondenti valori accelerometrici nelle tre direzioni x, y e z.

Pre-processing dei dati

Poiché l’obiettivo del nostro lavoro è quello di studiare solo il tremore a riposo sugli arti superiori, è stato necessario effettuare delle operazioni di pre-processing e filtraggio al fine di estrarre solo i dati utili per i nostri scopi. Per prima cosa dal file Annot di ogni paziente sono stati estratti solo i dati relativi alle prove di nostro interesse, identificati dalle etichette "updrs_3_17a" (RUE - Right Upper Extremity) e "updrs_3_17b" (LUE - Left Upper Extremity), ovvero le colonne 5 e 6.

Dopodiché, a partire da questi dati, è stato possibile ricavare i valori di accelerazione corrispondenti; i file utilizzati per estrarre le accelerazioni di ogni paziente sono denominati “rh” (Right anterior forearm) e “lh” (Left anterior forearm).

A questo punto, i valori estratti sono stati associati a quelli della scala UPDRS presenti sulla tabella ClinicData. Non è stato possibile studiare i casi di tremore posturale e cinetico in quanto il dataset non riportava i corrispondenti valori della scala UPDRS; dunque, non avremmo avuto alcun tipo di riscontro al termine della nostra analisi.

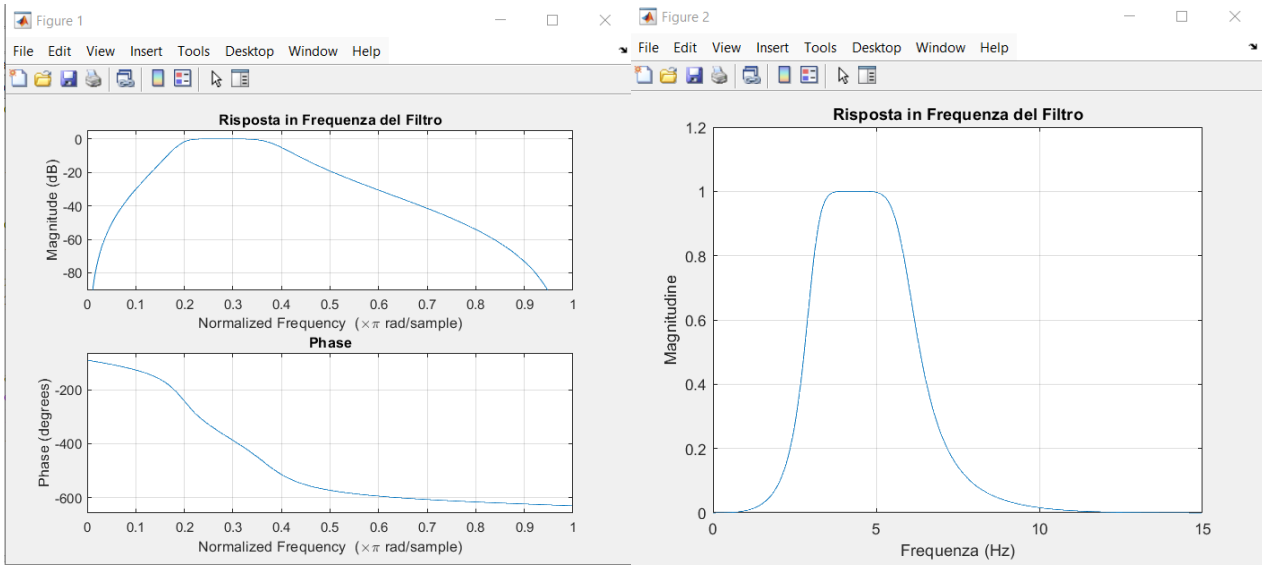
Per il pre-processing è stato realizzato uno script Matlab che per prima cosa ricerca l’evento target ‘UPDRS 3.17 – Rest Tremor Amplitude’ nel file Annot di ogni paziente e preleva tale riga e le due successive, contenenti le informazioni sulla medicazione e sulla parte del corpo. Poi i dati ricavati sono stati rielaborati e inseriti in una tabella di un’unica riga, in modo da rendere più agevole l’estrazione degli istanti temporali necessari per ottenere i valori accelerometrici corrispondenti a ciascuna prova.

I valori ottenuti sono stati inseriti in una tabella contenente l'indice della prova, gli istanti temporali, le accelerazioni, la medicazione e la parte del corpo. In seguito, grazie ad una funzione che estrae l'ID del paziente dal nome del file, è stato possibile associare ai dati accelerometrici contenuti nella tabella descritta in precedenza con i valori UPDRS corrispondenti, indicati nel file ClinicData. Il risultato finale di queste operazioni è una tabella unica contenente le prove di ogni paziente con i relativi dati accelerometrici, la durata e il livello nella scala UPDRS.

Classificatore basato su ML

Ricampionamento e Filtraggio

Dopo le operazioni di pre-processing del dataset per ottenere i dati utili per il nostro scopo, abbiamo proseguito ricampionando e filtrando i segnali di accelerazione.



A partire dalle informazioni presenti in letteratura abbiamo individuato la finestra di frequenze compresa tra 3Hz e 6Hz [2], [3] per effettuare il filtraggio relativo al tremore a riposo e abbiamo scelto di utilizzare un filtro di Butterworth passabanda di ordine 3. Per il filtraggio dei segnali abbiamo usato, come è possibile vedere sopra, un filtro passa-banda Butterworth di ordine 3 [4], [5]; questo perché un filtro di ordine più alto ha una pendenza maggiore nella banda di transizione (tra le frequenze di taglio) ma un ordine troppo elevato può portare a instabilità numeriche.

Quando si utilizza la funzione resample di Matlab per cambiare la frequenza di campionamento da 31.25 Hz a 30 Hz, il padding è necessario per ridurre al minimo gli artefatti e preservare la qualità del segnale interpolato. Questo avviene a causa delle trasformazioni numeriche intrinseche coinvolte nel ricampionamento.

La funzione resample utilizza un filtro antialiasing FIR (finite impulse response) per evitare aliasing quando si cambia la frequenza di campionamento; ciò introduce una latenza temporale e richiede un certo numero di campioni in ingresso per produrre un segnale stabile in uscita.

Poiché il filtro FIR necessità di un'ampia finestra di dati per calcolare le uscite, senza padding, esso non dispone di campioni sufficienti per la convoluzione, portando a instabilità o perdita di qualità; di conseguenza i bordi del segnale possono essere distorti, causando artefatti nella parte iniziale e finale

del segnale interpolato (infatti nel nostro segnale avevamo un picco anomalo rivolto verso l'asse delle ascisse nella parte finale del segnale).

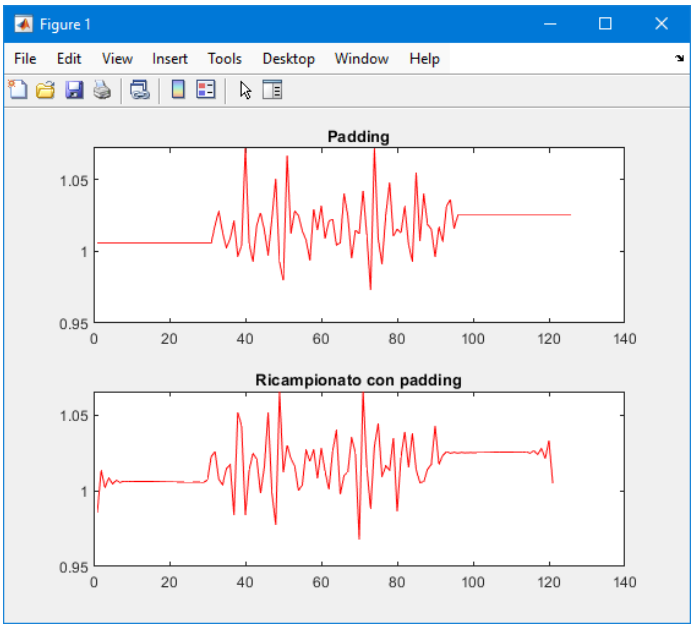
Inoltre, la lunghezza del filtro FIR dipende dai parametri di interpolazione (il rapporto tra frequenza di origine e di destinazione). Quando si passa da 31,25 Hz a 30 Hz, il rapporto richiede un filtro relativamente lungo, aumentando la necessità di padding per ridurre l'impatto del filtro.

Il padding aggiunge campioni fittizi (ad esempio, zeri o copie dei valori iniziali/finali del segnale) per ridurre gli effetti di bordo. Nel nostro caso abbiamo deciso di replicare i valori iniziali per la testa del segnale ed i valori finali per la coda del segnale e abbiamo aggiunto 30 campioni in testa e 30 campioni in coda, che vengono utilizzati solo durante il calcolo della convoluzione e poi rimossi alla fine.

Questa nostra scelta è stata dettata dall'esigenza di mantenere la continuità del segnale nei bordi, riducendo gli effetti di discontinuità rispetto al padding con zeri, ciò è importante per mantenere le proprietà visive nei processi di elaborazione di immagini e segnali (es. sfocatura, rilevamento bordi).

Il filtro FIR applicato durante resample utilizza i campioni aggiuntivi per elaborare correttamente il segnale vicino ai bordi. Questo produce un segnale interpolato più pulito, privo di distorsioni nei punti iniziali e finali.

Per fare padding abbiamo utilizzato la funzione 'repmat' che può essere usata per aggiungere zeri, bordi costanti o replicare i valori esistenti per espandere una matrice; nel nostro caso l'abbiamo preferita per semplificare l'implementazione del padding con valori uguali al bordo del segnale e per sfruttare le sue proprietà sopra descritte.



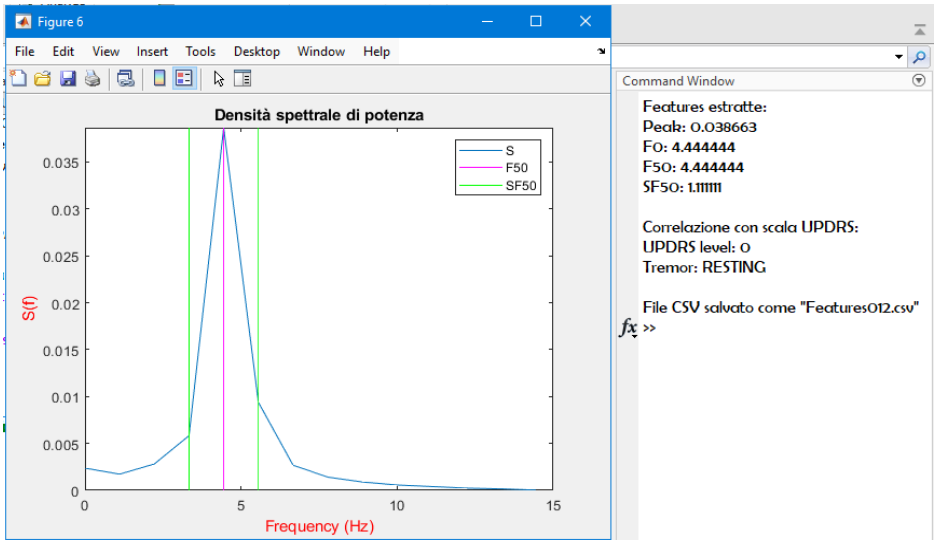
Dopodiché si è passati al calcolo della funzione di autocorrelazione e della densità spettrale di potenza, necessaria per l'operazione di estrazione delle features (descritta nel paragrafo successivo).

Estrazione delle features e stima del livello UPDRS

Le features sono attributi, proprietà o misurazioni derivate dai dati grezzi; l'estrazione delle features è un passaggio fondamentale che serve a trasformare i dati in una rappresentazione più informativa e utile per l'addestramento dei modelli di Machine Learning.

Le features sono state estratte a partire dalla trasformata di Fourier (FFT) del vettore accelerazione campionato e filtrato, ovvero la densità spettrale di potenza (PSD), che rappresenta la distribuzione della potenza di un segnale nelle diverse frequenze. Le features principali sono riportate di seguito:

1. **Peak**: il valore massimo della densità spettrale di potenza, cioè l'ampiezza del picco principale
2. **Fo**: la frequenza associata al picco massimo, cioè la frequenza dominante, quella con il contenuto energetico maggiore nel segnale
3. **F50**: la frequenza mediana sotto la quale è contenuto il 50% della potenza totale del segnale
4. **SF50**: la larghezza dello spettro attorno alla frequenza F50, cioè la banda di frequenze che contiene il 68% della potenza totale del segnale.



La figura sopra rappresenta il plot della densità spettrale di potenza del modulo dell'accelerazione filtrato, relativo ad un singolo paziente, dal quale sono state estratte le 4 features. Dalla figura si osservano: in blu il picco principale; in rosso la sua frequenza corrispondente Fo, che coincide con la frequenza mediana F50; in verde l'intervallo SF50 centrato su F50.

Oltre alle 4 features sopra, sono state estratte anche altre features, tramite le corrispondenti funzioni MATLAB applicate allo spettro del segnale, per ottenere i principali parametri statistici: il valore massimo e minimo, la media, la deviazione standard, i parametri che misurano l'asimmetria dello spettro (skewness) e l'appiattimento (curtosi), la mediana degli scostamenti dello spettro, l'ampiezza picco-picco, la moda e il valore massimo dei picchi.

È stata creata una funzione apposita ("*features_extraction*") che prende in input i vettori delle accelerazioni lungo i 3 assi x, y e z per ciascuna prova di ogni paziente e restituisce come output le 4 features principali e un vettore di features aggiuntive, in modo da facilitare la selezione delle features desiderate. Questa funzione, oltre al calcolo delle features, include anche il pre-processing dei vettori accelerazione (illustrato nel paragrafo precedente).

In aggiunta, è stata realizzata la funzione ("*stima_UPDRS*") che stima il livello UPDRS basandosi sul valore del picco dello spettro e classifica il tipo di tremore del paziente (Resting, Postural, Kinetic) in

base al valore della frequenza Fo corrispondente al picco principale. Il livello UPDRS viene assegnato in base all'intervallo della variabile peak:

- `peak < 5` → Livello 0
- `5.001 < peak < 32` → Livello 1
- `32.001 < peak < 200` → Livello 2
- `200.001 < peak < 300` → Livello 3
- `peak > 300.001` → Livello 4

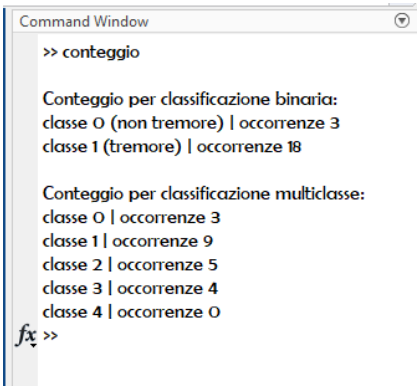
L'obiettivo è il confronto tra questa stima basata su soglie statiche, rispetto al valore che si ottiene con il ML, utilizzando come input le features estratte.

Entrambe le funzioni sono state inserite nel codice principale ("*ALL_features*"), il quale itera su ciascun paziente e su ciascun trial (RUE off/on, LUE off/on) per estrarre i 3 vettori di accelerazione dai quali ricavare le features. Il numero di pazienti esaminati è 12, per un totale di 21 prove; al termine del pre-processing il numero di prove esaminate era 23 ma due di queste, relative ai pazienti con ID 006 e 035, sono state scartate in quanto di durata troppo breve, poichè ciò rendeva impossibile l'operazione di filtraggio.

I dati sono stati raccolti in una tabella in formato CSV, di 21 righe e 20 colonne. Le righe corrispondono alle prove totali per tutti i pazienti sotto esame, le colonne invece sono organizzate come segue: le prime 14 contengono i valori delle features estratte, poi c'è l'identificativo del paziente, il numero e il tipo di prova, il valore della scala UPDRS reale (*Label_UPDRS*) e quello stimato dall'algoritmo (*Stima_UPDRS*), infine è stata aggiunta una colonna per la classificazione binaria. Quest'ultima, partendo dal valore UPDRS reale, assegna l'etichetta 0 in caso di non tremore, 1 per i livelli di tremore 1/2/3/4. Ciò potrebbe essere utile a causa della scarsità dei dati estratti: classificando in base a tremore/non tremore, l'accuratezza dovrebbe migliorare rispetto alla classificazione per livelli.

Un primo problema riscontrato è che i valori dei picchi sono tutti molto bassi e inferiori alla prima soglia che è a 5, quindi tutti i valori UPDRS stimati con l'algoritmo a soglie è 0, tranne quello del primo paziente, l'unico appartenente alla classe 1, poiché ha un picco di ampiezza maggiore di 5.

Un altro problema è che i dati sono pochi e il dataset è sbilanciato; la distribuzione di ogni classe, sia per classificazione binaria (0-1) che multiclasse (0-1-2-3-4), è la seguente:



Per bilanciare le classi, quindi, è necessario effettuare data augmentation tramite la funzione SMOTE di MATLAB, che risolve questo problema generando nuovi esempi sintetici per le classi minoritarie.

Un altro passaggio fondamentale è la normalizzazione delle features, che serve a scalare tutti i valori in un intervallo specifico, per migliorare le prestazioni del modello e facilitare il confronto tra diverse features.

Normalizzazione

La normalizzazione è un passaggio chiave per garantire che i dati siano ben preparati per l'addestramento del modello, inoltre aiuta a migliorare l'accuratezza, la stabilità e la velocità di convergenza, garantendo che tutte le caratteristiche contribuiscano equamente alla predizione. La normalizzazione trasforma i dati per scalare i valori numerici in un intervallo predefinito, tipicamente [0,1] o [−1,1], nello specifico nel nostro caso abbiamo adottato il range [0,1]. Ad esempio, se un dataset ha caratteristiche numeriche che variano tra x_{min} e x_{max} , la normalizzazione Min-Max ridimensiona ogni valore x secondo la formula:

$$xnorm = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Benefici della normalizzazione:

- Migliora la prestazione degli algoritmi: la normalizzazione permette agli algoritmi di trattare tutte le caratteristiche in modo equilibrato, evitando di enfatizzare alcune a scapito di altre e questo può portare a un'accuratezza maggiore.
- Stabilità numerica: gli algoritmi che eseguono molte iterazioni di calcolo (es., le reti neurali) possono diventare instabili se i valori in input hanno scale molto diverse e la normalizzazione riduce il rischio di overflow o underflow numerico.
- Intervallo uniforme per interpretabilità: trasformando i dati in un intervallo uniforme ([0,1][0,1][0,1]), diventa più facile analizzare le distribuzioni e confrontare le caratteristiche.
- Migliora la generalizzazione: quando i dati sono normalizzati, gli algoritmi tendono a generalizzare meglio, cioè a funzionare bene sia sul training set che sui dati mai visti prima.

Modelli di classificazione utilizzati

Le feature estratte e normalizzate sono state utilizzate per realizzare la Binary Classification e la Multilabel Classification. La Multilabel Classification considera 4 label (level UPDRS:0,1,2,3) e in entrambe le classificazioni è stata utilizzata la “Synthetic Minority Over-sampling Technique” perché la distribuzione dei campioni è molto sbilanciata.

Non adottare la tecnica “SMOTE” non permetterebbe al modello (machine learning) di imparare correttamente la classe minoritaria, perché è più esposto ai campioni della classe maggioritaria. Per cui durante l'allenamento il modello tenderà a prevedere principalmente la classe che ha più esempi (la classe maggioritaria), ignorando o facendo poche previsioni per la classe minoritaria.

SMOTE (Synthetic Minority Over-sampling Technique)

È una tecnica di oversampling utilizzata nel campo del machine learning per affrontare il problema dello squilibrio nelle classi di un dataset. Quando un dataset contiene una classe minoritaria, cioè rappresentata da pochi campioni rispetto a un'altra, SMOTE viene utilizzato per aumentare il numero di campioni della classe minoritaria attraverso la generazione di campioni sintetici. Ciò avviene

prendendo un campione della classe minoritaria ed uno dei suoi vicini (scelto in maniera casuale). Il campione sintetico viene creato lungo la linea che collega i due punti, creando un campione che ha caratteristiche simili ma non identiche al campione originale.

Questa tecnica è stata utilizzata considerando la funzione: "mySMOTE" reperibile nel sito:

https://it.mathworks.com/matlabcentral/fileexchange/75401-synthetic-minority-over-sampling-technique-smote

I campioni generati dalla funzione mySMOTE si trovano nello stesso spazio delle caratteristiche (spazio delle feature) in cui si trovano i campioni originali. La stessa funzione è stata utilizzata in entrambi i modelli (nel caso della MultilabelClassification, la funzione è stata iterata (con un ciclo for)).

A questa funzione è stato dato in input:

- Il dataset composto dalle feature normalizzate e dalla label contenente il livello UPDRS 0, 1 per il BinaryClassification (0,1,2,3 per il MultilabelClassification).
- L'etichetta (le etichette) della classe/i minoritarie.
- un numero intero che indica quanti nuovi dati sintetici generare.

La scelta del numero è stata fatta tenendo conto dell'OVERFITTING (è un fenomeno che si verifica quando un modello di machine learning impara troppo bene i dettagli e il rumore del dataset di addestramento. Questo significa che il modello non solo cattura le relazioni genuine tra le variabili, ma memorizza anche fluttuazioni casuali o rumore nei dati).

Classification Learner

Classification Learner è un'applicazione di MATLAB che consente di esplorare facilmente vari algoritmi di classificazione e addestrarli su un set di dati, tramite un'interfaccia grafica.

Gli algoritmi di classificazione sono tecniche di machine learning (di sviluppo di algoritmi che permettono ai computer di apprendere dai dati e di prendere decisioni senza essere esplicitamente programmati per farlo) utilizzate per predire una categoria o classe per un dato di input, in base alle sue caratteristiche. Questi algoritmi vengono applicati a problemi di classificazione, dove l'obiettivo è associare un'etichetta a ciascun dato in ingresso) e addestrarli su un set di dati.

Nell'applicazione è stato caricato il dataset delle due classificazioni dal workspace di MATLAB e il tool ha identificato automaticamente le colonne delle feature e quella delle etichette (leve UPDRS); poi è stato possibile, attraverso l'opzione "ALL", allenare tutti i modelli nell'app, in questa maniera MATLAB addestra i vari modelli di classificazione e li confronta in modo automatico.

Interpretazione dei risultati ottenuti dal Classification Learner

Sono state scelte due modalità di valutazione dei risultati ottenuti dagli algoritmi di classificazione:

- **Matrice di Confusione**: Una matrice di confusione è una tabella quadrata, generalmente 2x2 per problemi di classificazione binaria, ma può essere di dimensioni maggiori (NxN) per problemi di classificazione multiclasse.
    - **Interpretazione**: sulla **diagonale principale** ho le classificazioni corrette, nella off-diagonal ho le classificazioni sbagliate (gli errori di classificazione, ovvero le occorrenze

in cui il modello ha fatto previsioni sbagliate). Ad esempio, se un modello ha una matrice di confusione con valori alti sulla diagonale e bassi fuori di essa, significa che il modello sta facendo previsioni corrette per la maggior parte dei dati.
    - **Utilità**: È utile per capire il tipo di errori che il modello sta commettendo (ad esempio, se sta confondendo alcune classi).
- L'**accuratezza** è una delle metriche più comuni per valutare un modello di classificazione poiché esprime la percentuale di predizioni corrette.
    - **Interpretazione**: Un'accuratezza alta (vicina al 100%) indica che il modello sta facendo un buon lavoro. Tuttavia, se il dataset è sbilanciato (es. una classe è molto più frequente dell'altra), un'accuratezza alta può non essere indicativa di un buon modello.
    - **Limitazioni:** Può essere fuorviante quando il dataset è sbilanciato, cioè quando una classe è molto più rappresentata rispetto ad un'altra, poiché in questi casi, un modello che predice sempre la classe più frequente può ottenere una buona accuratezza senza effettivamente fare un buon lavoro nel riconoscere la classe minoritaria.

Risultati

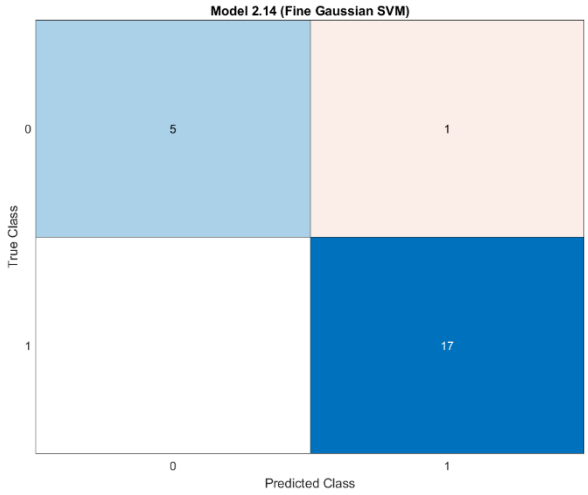
Risultati Binary classification

Model Num...	Model Type	Status	Accuracy (Validation)
1	Tree	Trained	47.83 %
2.1	Tree	Trained	47.83 %
2.2	Tree	Trained	47.83 %
2.3	Tree	Trained	47.83 %
2.4	Discriminant	Trained	47.83 %
2.5	Discriminant	Failed	-
2.6	Binary GLM Logistic Regression	Trained	69.57 %
2.7	Efficient Logistic Regression	Trained	73.91 %
2.8	Efficient Linear SVM	Trained	73.91 %
2.9	Naive Bayes	Trained	73.91 %
2.10	Naive Bayes	Trained	86.96 %
2.11	SVM	Trained	69.57 %
2.12	SVM	Trained	91.30 %
2.13	SVM	Trained	86.96 %
2.14	SVM	Trained	95.65 %
2.15	SVM	Trained	78.26 %
2.16	SVM	Trained	73.91 %
2.17	KNN	Trained	78.26 %
2.18	KNN	Trained	52.17 %
2.19	KNN	Trained	73.91 %
2.20	KNN	Trained	60.87 %
2.21	KNN	Trained	73.91 %
2.22	KNN	Trained	82.61 %
2.23	Ensemble	Trained	73.91 %
2.24	Ensemble	Trained	65.22 %
2.25	Ensemble	Trained	65.22 %
2.26	Ensemble	Trained	78.26 %
2.27	Ensemble	Trained	56.52 %
2.28	Neural Network	Trained	78.26 %
2.29	Neural Network	Trained	82.61 %
2.30	Neural Network	Trained	78.26 %
2.31	Neural Network	Trained	73.91 %
2.32	Neural Network	Trained	86.96 %
2.33	Kernel	Trained	91.30 %
2.34	Kernel	Trained	73.91 %

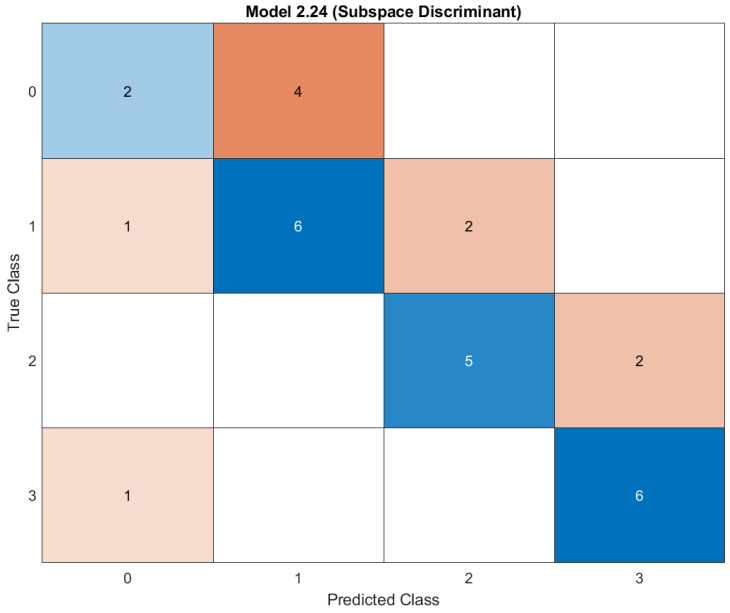
Dai risultati ottenuti si ha che per la Binary classification l'algoritmo di classificazione che fa previsioni più corrette è dato dal "Model 2.14": il modello SVM (Support Vector Machine) è in grado di classificare

correttamente una percentuale maggiore di dati rispetto ad altri modelli testati, con un'accuracy che raggiunge il **95.65%**.

Dalla matrice di confusione si osserva un buon addestramento, sbagliando solo per un campione:



Risultati Multilevel classification



Model Num...	Model Type	Status	Accuracy (Validation)
1	Tree	Trained	48.28 %
2.1	Tree	Trained	48.28 %
2.2	Tree	Trained	48.28 %
2.3	Tree	Trained	48.28 %
2.4	Discriminant	Trained	55.17 %
2.5	Discriminant	Failed	-
2.6	Efficient Logistic Regression	Trained	31.03 %
2.7	Efficient Linear SVM	Trained	31.03 %
2.8	Naive Bayes	Trained	51.72 %
2.9	Naive Bayes	Trained	48.28 %
2.10	SVM	Trained	62.07 %
2.11	SVM	Trained	58.62 %
2.12	SVM	Trained	55.17 %
2.13	SVM	Trained	51.72 %
2.14	SVM	Trained	55.17 %
2.15	SVM	Trained	31.03 %
2.16	KNN	Trained	51.72 %
2.17	KNN	Trained	31.03 %
2.18	KNN	Trained	31.03 %
2.19	KNN	Trained	44.83 %
2.20	KNN	Trained	24.14 %
2.21	KNN	Trained	48.28 %
2.22	Ensemble	Trained	31.03 %
2.23	Ensemble	Trained	51.72 %
2.24	Ensemble	Trained	65.52 %
2.25	Ensemble	Trained	51.72 %
2.26	Ensemble	Trained	48.28 %
2.27	Neural Network	Trained	48.28 %
2.28	Neural Network	Trained	48.28 %
2.29	Neural Network	Trained	44.83 %
2.30	Neural Network	Trained	55.17 %
2.31	Neural Network	Trained	51.72 %
2.32	Kernel	Trained	58.62 %
2.33	Kernel	Trained	51.72 %

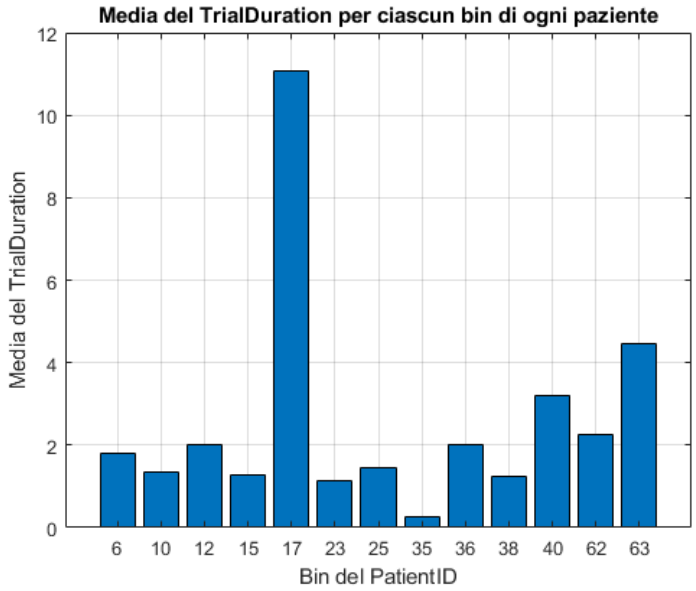
Nel caso del Multilevel classification, l'algoritmo di classificazione che fa previsioni più corrette è dato dal "Model 2.24", con un'accuracy del **65.52%**, in cui viene utilizzato un algoritmo Ensemble, cioè una tecnica di apprendimento automatico che combina più modelli di base per ottenere una previsione più accurata e robusta rispetto all'utilizzo di un singolo modello.

Dalla matrice di confusione si osserva che l'algoritmo di classificazione non apprende bene il livello UPDRS 0 visto che viene equivocato con il livello UPDRS 1.

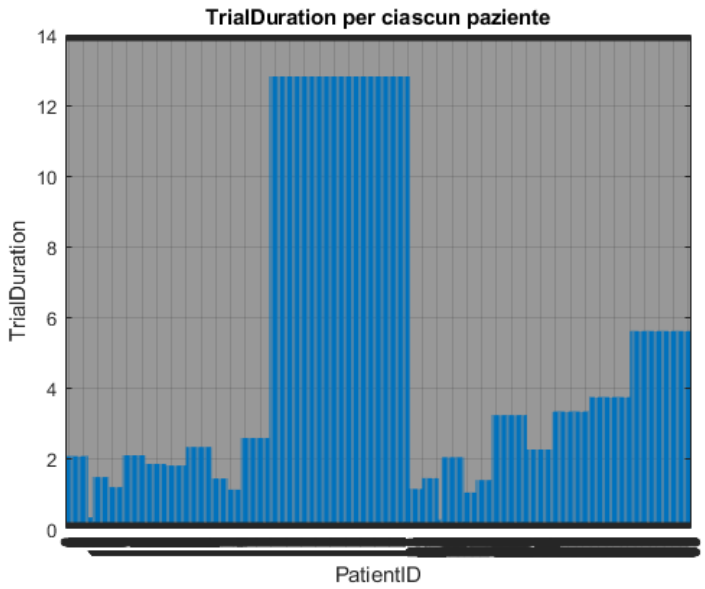
Dopo aver applicato la correlazione tra features per capire quali fossero le più utili da considerare, e utilizzando il modello Tree si è riuscito ad ottenere un'accuracy del **78.4%.**

Considerazioni grafiche dei dati

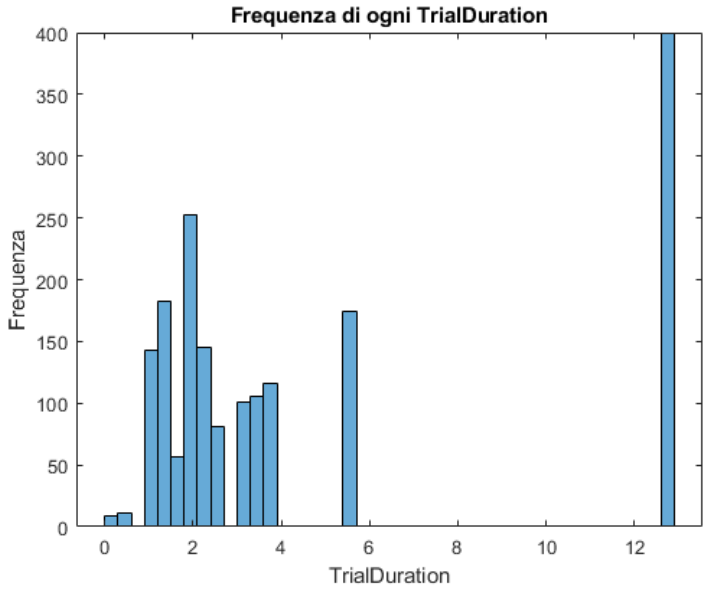
Grafici per la tabella estratta dal dataset originale



(1)



(2)

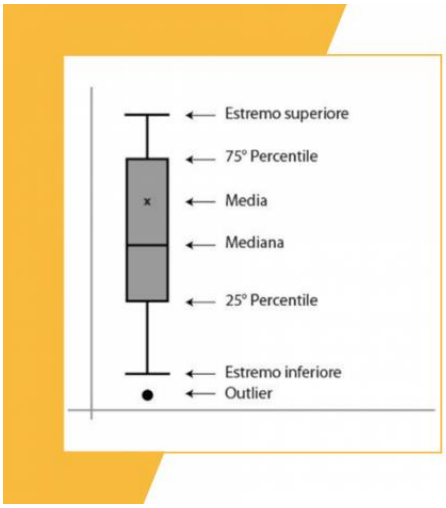


(3)

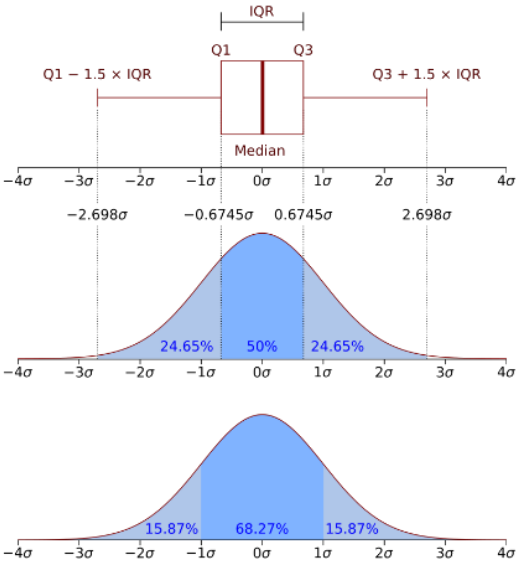
(2) e (3) sono serviti per capire la distribuzione dei valori di TrialDuration, e applicare poi la finestratura, la quale prevede l'eliminazione delle prove sotto a 1 s: infatti nella tabella estratta dal dataset originale sono stati considerati tutti i pazienti e tutte le prove, precisamente 13 pazienti e 23 prove, mentre per la tabella delle features (cioè dopo aver applicato la finestratura) sono stati presi in considerazione 12 pazienti e 21 prove complessivamente.

Per ogni singola prova, i valori di TrialDuration sono stati ricavati facendo la differenza tra l'ultimo e il primo istante di Timestamp.

Diagramma a scatola e baffi (boxplot)



(4)



(5)

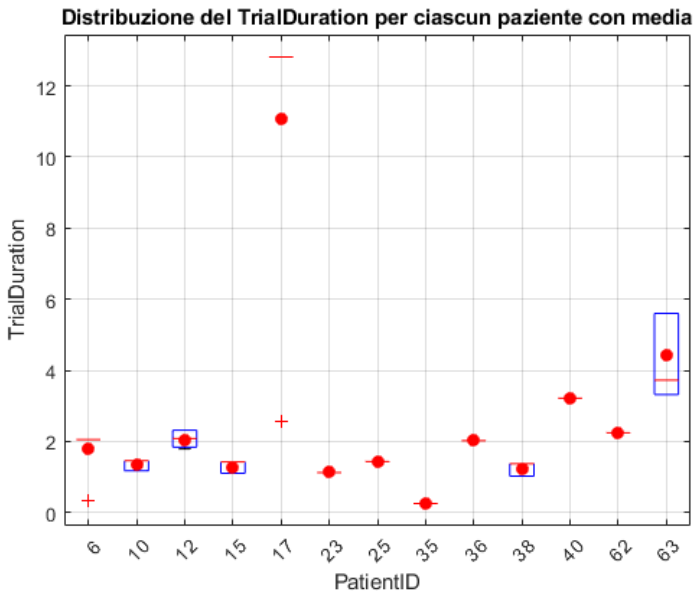
(4) e (5) mostrano il significato di boxplot, cioè una rappresentazione grafica utilizzata per descrivere la distribuzione di un campione mediante l'uso di indici di dispersione e di posizione:

- Q1 = 1° quartile (25%)
- Q3 = 3° quartile (75%)
- media (in generale non coincide con la mediana)
- mediana = 2° quartile (50%)
- estremo superiore = valore massimo
- estremo inferiore = valore minimo
- outlier = valori fuorvianti, numericamente distanti dal resto dei valori presenti nel campione
- IQR (scarto interquartile) = Q3-Q1

La lunghezza dei baffi (segmenti tra i 2 quartili e gli estremi) si può estendere fino ad un massimo di:

- Q1 - 1.5 x IQR (sotto il 1° quartile)
- Q3 + 1.5 x IQR (sopra il 3° quartile)

Tutti i valori esterni rispetto a tali distanze sono ritenuti valori outlier.



(6) - riferimento alla tabella estratta dal dataset originale

Come si enuncia da (6) non è stato possibile ottenere dei boxplot completi, visto che ci sono numerosi valori di TrialDuration che risultano costantemente uguali per un dato PatientID. Per ottenere dei boxplot completi, avremmo avuto bisogno di una variazione più corposa dei dati (almeno 2 valori distinti per ogni paziente).

Gli outlier sono rappresentati tramite “+” rossi.

Valori della media sono rappresentati mediante pallini rossi.

Valori della mediana rappresentati attraverso linee rosse orizzontali.

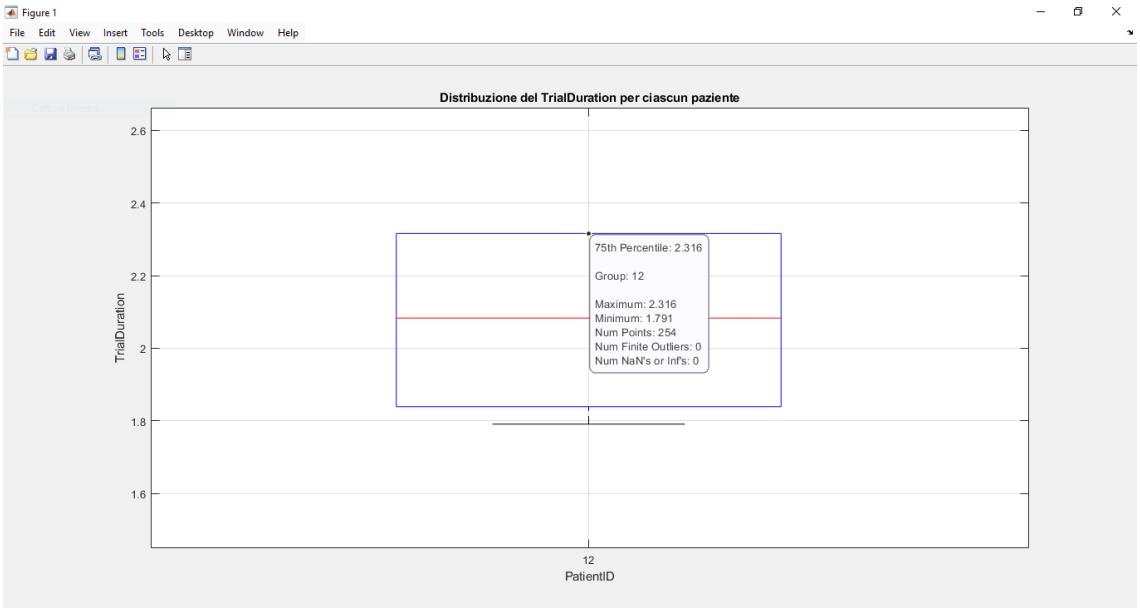


Figure 1
File Edit View Insert Tools Desktop Window Help
Distribuzione del TrialDuration per ciascun paziente
TrialDuration
2.6
2.4
2.2
2
1.8
1.6
75th Percentile: 2.316
Group: 12
Maximum: 2.316
Minimum: 1.791
Num Points: 254
Num Finite Outliers: 0
Num NaN's or Inf's: 0
12
PatientID

(7)

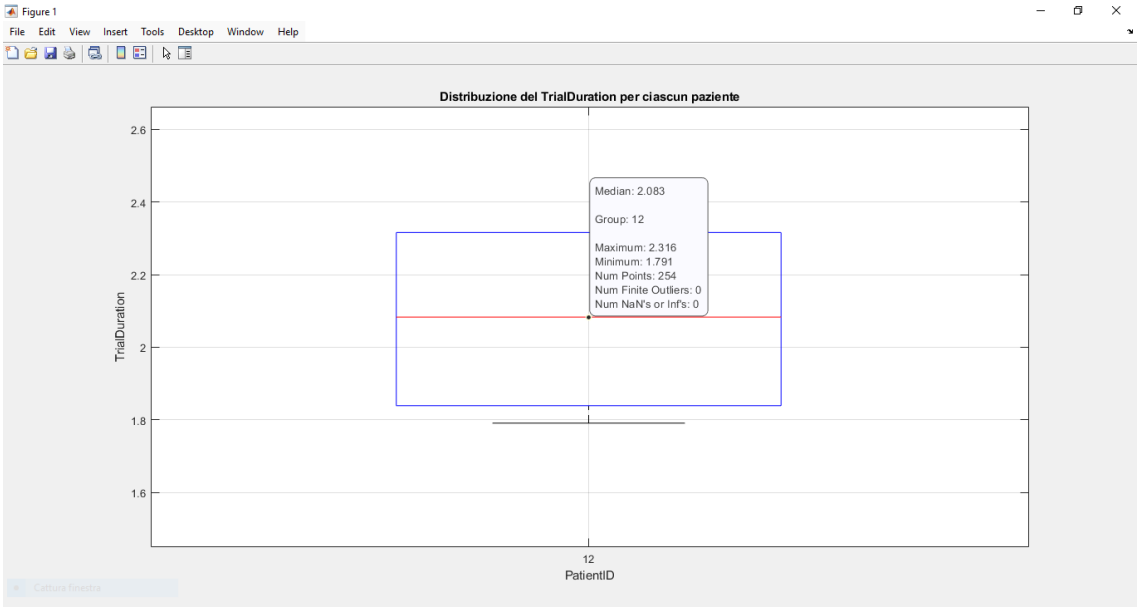


Figure 1
File Edit View Insert Tools Desktop Window Help
Distribuzione del TrialDuration per ciascun paziente
TrialDuration
2.6
2.4
2.2
2
1.8
1.6
Median: 2.083
Group: 12
Maximum: 2.316
Minimum: 1.791
Num Points: 254
Num Finite Outliers: 0
Num NaN's or Inf's: 0
12
PatientID

(8)

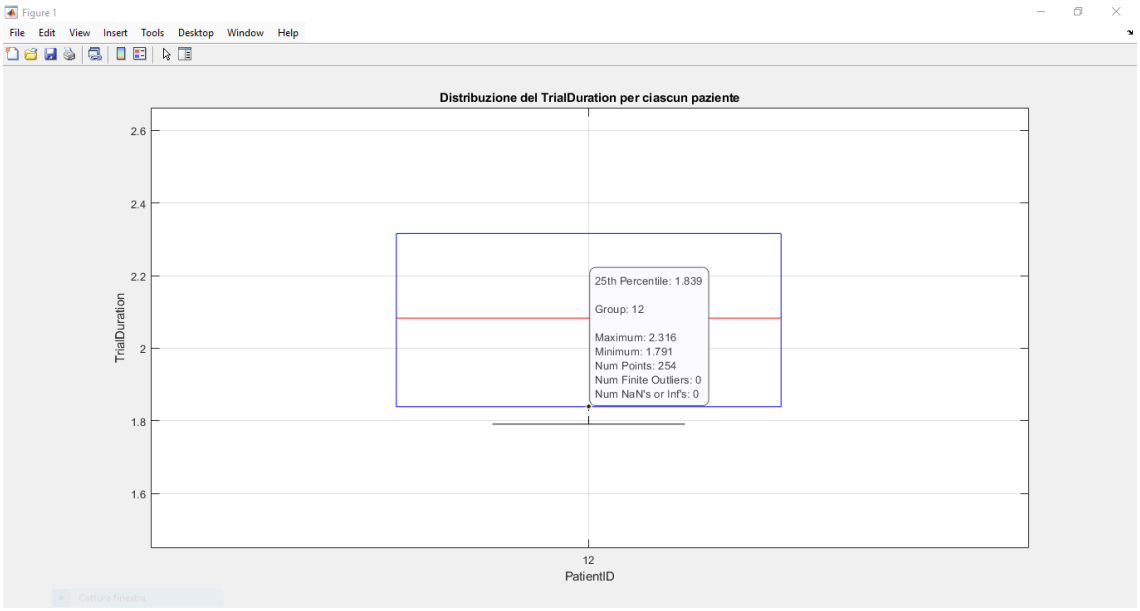


Figure 1
File Edit View Insert Tools Desktop Window Help
Distribuzione del TrialDuration per ciascun paziente
TrialDuration
2.6
2.4
2.2
2
1.8
1.6
25th Percentile: 1.839
Group: 12
Maximum: 2.316
Minimum: 1.791
Num Points: 254
Num Finite Outliers: 0
Num NaN's or Inf's: 0
12
PatientID

(9)

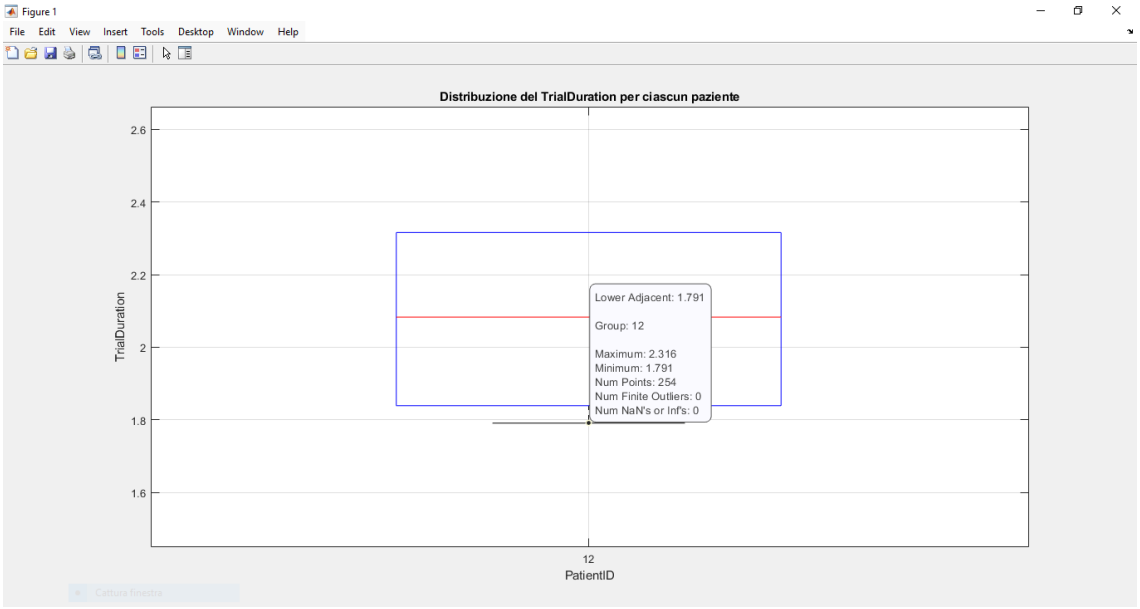
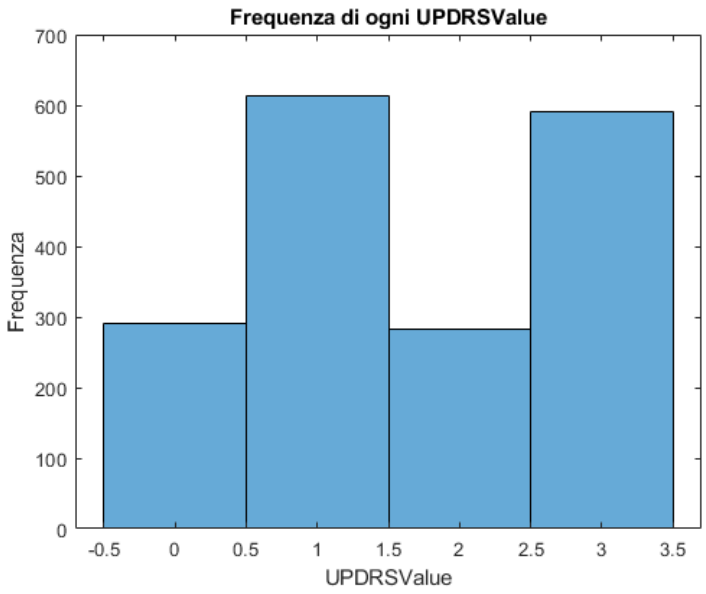
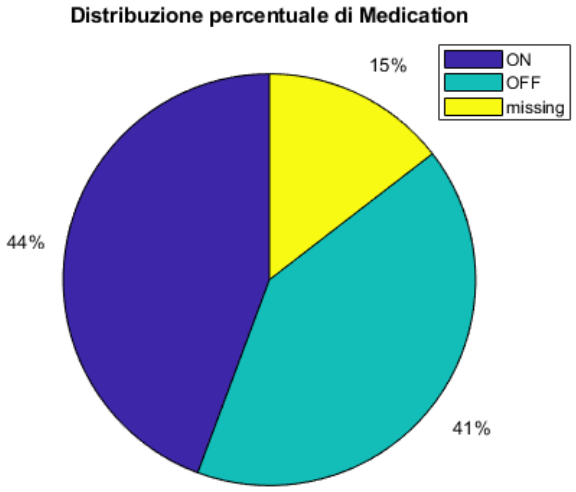


Figure 1
File Edit View Insert Tools Desktop Window Help
Distribuzione del TrialDuration per ciascun paziente
TrialDuration
2.6
2.4
2.2
2
1.8
1.6
Lower Adjacent: 1.791
Group: 12
Maximum: 2.316
Minimum: 1.791
Num Points: 254
Num Finite Outliers: 0
Num NaN's or Inf's: 0
12
PatientID

(10)



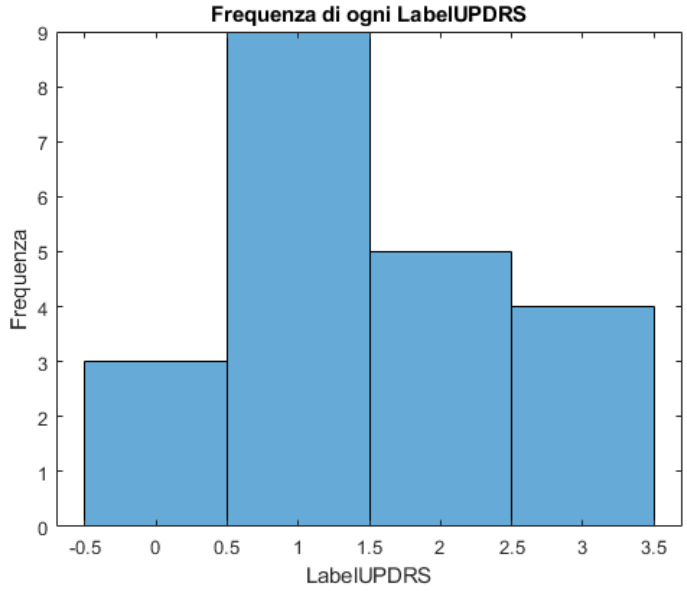
(11)



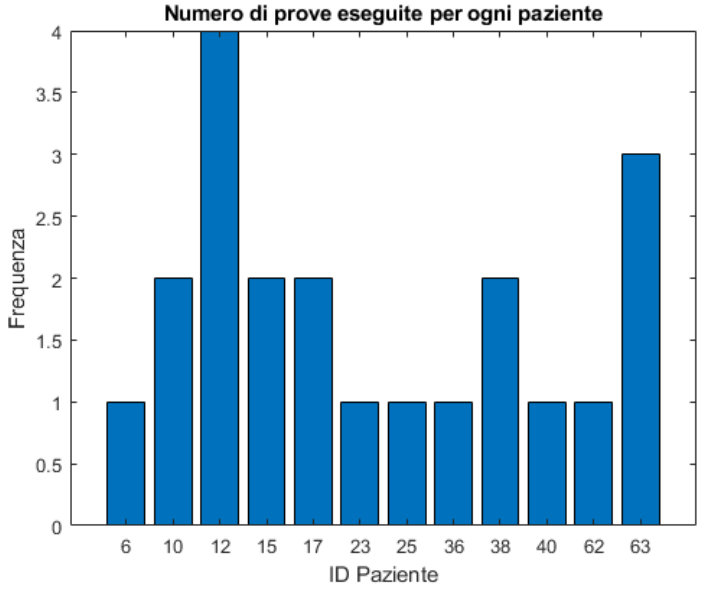
(12)

(11) è risultato fondamentale per chiarire le idee sulla frequenza di UPDRSValue: si può notare che i valori 1 e 3 sono quelli più assidui, e risultano sbilanciati se confrontati con 0 e 2. Per poter rimediare a tale problema viene fatto capo alla funzione “Synthetic Minority Over-sampling Technique” (SMOTE), utile appunto per poter bilanciare tra loro tali valori.

Grafici per la tabella delle features



(13)



(14)

Bibliografia

[1] J. L. Adams, K. Dinesh, C. W. Snyder, M. Xiong, C. G. Tarolli, S. Sharma, E. R. Dorsey e G. Sharma, «A real-world study of wearable sensors in Parkinson's disease,» *npj Parkinson's Disease,* pp. 1-8, 2021.

[2] L. Sigcha, I. Pavon, N. Costa, S. Costa, M. Gago, P. Arezes, J. M. Lopez e G. De arcas, «Automatic Resting Tremor Assessment in Parkinson's Disease Using Smartwatches and Multitask Convolutional Neural Networks,» *Sensors,* pp. 21(1), 291, 2021.

[3] Y. Zhao, X. Wang, X. Peng, Z. Li, F. Nan, M. Zhou, J. Qi, Y. Yang, Z. Zhao, L. Xu e P. Yang, «Selecting and Evaluating Key MDS-UPDRS,» *IEEE JOURNAL OF SELECTED AREASIN SENSORS, VOL. 1,* pp. 177-188, 2024.

[4] C. Crowe, M. Sica, L. Kenny, B. O'Flynn, D. S. Mueller, S. Timmons, J. Barton e S. Tedesco, «Wearable-Enabled Algorithms for the Estimation,» *IEEE TRANSACTIONS ON NEURAL SYSTEMS ANDREHABILITATION ENGINEERING, VOL. 32,* pp. 3028-3036, 2024.

[5] R. San-Segundo, A. Zhang, A. Cebulla, S. Panev, G. Tabor, K. Stebbins, R. E. Massa, A. Whitford, F. de la Torre e J. Hodgins, «Parkinson's Disease Tremor Detection in the Wild Using Wearable Accelerometers,» *Sensors,* pp. 20(20), 5817, 2020.