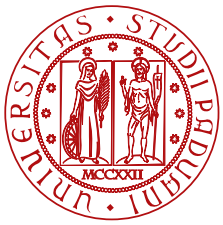


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Implementazione di Algoritmi Predittivi per l'Assistenza Proattiva in Deambulatori Tramite Sensori di Pressione a Basso Costo

Relatore
Prof. Luca Tonin

Laureando
Lorenzo Croce

Correlatore
Luca Zanchi

ANNO ACCADEMICO 2023-2024

Data di Laurea 26/09/2024

Raggiungere questo traguardo è stato possibile grazie all'affetto, al sostegno e all'incoraggiamento di alcune persone straordinarie che hanno reso il mio cammino più ricco e significativo.

Ai miei genitori, Claudio e Letizia, per avermi sempre trasmesso l'importanza dell'impegno e della dedizione, e per essere stati il mio punto di riferimento costante. Grazie per avermi spronato a credere in me stesso, rendendo ogni passo più sicuro.

Alle mie nonne, Concetta e Anna, che con il loro amore incondizionato mi hanno insegnato il valore del sostegno familiare. La vostra fiducia in me ha sempre acceso la mia motivazione.

A tutta la famiglia di ArenaCraft, che mi ha accolto, supportato, sopportato e dato l'opportunità di crescere non solo professionalmente, ma anche umanamente. Grazie per essere stati al mio fianco in questo percorso di arricchimento.

Infine, al mio amico Alberto, per il suo ineguagliabile sostegno durante le sfide dello studio e per la mano tesa nei momenti di difficoltà. La tua amicizia è stata una risorsa preziosa che non dimenticherò mai.

A tutti voi, questo traguardo è anche il vostro successo. Grazie di cuore.

Sommario

Nel primo capitolo viene discusso come le pelli sensoriali artificiali, utilizzate nella robotica e nell'interazione uomo-macchina, replichino la percezione tattile umana per migliorare l'interazione dei robot con l'ambiente. Esistono vari tipi di sensori tattili, come quelli piezoresistivi, capacitivi e piezoelettrici, ognuno basato su principi diversi per rilevare pressione e sollecitazioni. Nonostante le loro applicazioni promettenti, come nel monitoraggio fisiologico e nella manipolazione di oggetti, la produzione su larga scala è ancora complessa e costosa. Questa tesi si propone di sviluppare algoritmi per riconoscere i movimenti utilizzando sensori tattili a basso costo, rendendoli accessibili per un uso più ampio.

Il secondo capitolo descrive l'utilizzo di un sensore di pressione piezoresistivo selezionato per il suo buon rapporto qualità-prezzo, che sfrutta la variazione di resistenza del materiale dielettrico in risposta a una forza. Il sensore è costituito da strisce di rame disposte perpendicolarmente su un substrato plastico dielettrico, permettendo la rilevazione della pressione in punti distanziati di 40 mm. Questo sensore è stato utilizzato per addestrare diversi modelli di classificazione per il riconoscimento di cinque movimenti, con l'obiettivo di prevedere l'intenzione dell'utente. Per la raccolta dei dati, è stato sviluppato un software in Python con un'interfaccia intuitiva che guida l'utente nella registrazione dei movimenti. I dati acquisiti vengono organizzati e salvati in formato JSON per un facile utilizzo nell'addestramento dei modelli.

Nel terzo capitolo si analizzano le prestazioni del sensore. I dati raccolti mostrano una griglia di valori che riflettono l'intensità del voltaggio della corrente in risposta alla pressione applicata. I risultati dell'addestramento dei modelli di classificazione sui movimenti campione rivelano che, tra i vari algoritmi testati, il modello Random Forest ha ottenuto la migliore accuratezza. Vengono inoltre visualizzati graficamente i pesi delle feature del Random Forest dopo l'addestramento.

In conclusione il sensore ha dimostrato di rilevare correttamente la pressione e di fornire dati affidabili, permettendo una buona classificazione dei movimenti dell'utente, soprattutto con il modello Random Forest. Tuttavia, la lentezza nella classificazione, che richiede 7 secondi di dati, limita l'uso in scenari dove è necessaria una risposta in tempo reale. La durata dell'acquisizione non è stata ottimizzata e necessita di ulteriori test per ridurla senza compro-

mettere l’accuratezza. Inoltre, il sistema deve essere testato su utenti con reali problemi motori e ottimizzato per adattarsi meglio alle caratteristiche fisiche degli utilizzatori.

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Background e stato dell'arte

Le pelli sensoriali artificiali, o sensori tattili, trovano impiego in ambiti quali l'interazione uomo-macchina e la robotica. Questi sensori sono progettati per replicare la percezione tattile umana, permettendo ai dispositivi e ai robot di "sentire" e rispondere ai contatti fisici con l'ambiente circostante. La capacità di rilevare variazioni di pressione, forza e altre sollecitazioni meccaniche consente a queste tecnologie di migliorare l'interazione e la funzionalità dei sistemi automatizzati, offrendo una vasta gamma di applicazioni in dispositivi indossabili, protesi e robot autonomi. In questa sezione, vengono discusse le diverse tipologie di sensori tattili, ciascuna con specifiche caratteristiche e principi di funzionamento.

- **Sensori piezoresistivi:** Questi sensori sfruttano l'effetto piezoresistivo ovvero la proprietà di alcuni materiali di cambiare la loro resistenza elettrica quando sottoposti a stress meccanico, come la pressione. [1] [2]

- **Sensori capacitivi:** Questi sensori misurano la variazione di capacità elettrica per rilevare la pressione o la prossimità di un oggetto. Un sensore capacitivo di base è costituito da due piastre conduttive separate da un materiale isolante detto dielettrico [1]

- **Sensori piezoelettrici:** Sfruttano l'effetto piezoelettrico per misurare la pressione, le vibrazioni o altre grandezze fisiche che possono essere convertite in una forza. L'effetto piezoelettrico è la capacità di alcuni materiali di generare una carica elettrica quando sottoposti a stress meccanico. Questa carica elettrica è proporzionale alla forza applicata al materiale.[3]

La scelta del materiale per un sensore tattile dipende da diversi fattori, tra cui la sensibilità richiesta, la gamma di pressione da misurare, la flessibilità e l'estensibilità del sensore e il costo.

1.2 Applicazioni della classificazione dei segnali

La classificazione dei segnali raccolti dai sensori ha una vasta gamma di applicazioni pratiche che spaziano dalla robotica all'assistenza sanitaria, fino all'interazione uomo-robot.

1.2.1 Robotica

Nel campo della robotica, la classificazione dei segnali tattili raccolti dai sensori riveste un'importanza cruciale. Queste pelli artificiali, spesso realizzate con matrici di sensori piezoelettrici, capacitivi o basati su tomografia a impedenza elettrica (EIT), sono progettate per permettere ai robot di interagire con l'ambiente in modo più naturale e sicuro. I sensori integrati raccolgono segnali tattili che vengono poi classificati per determinare la posizione, l'intensità e la natura del contatto. [4] [5].

Inoltre, la classificazione dei segnali tattili gioca un ruolo essenziale nel migliorare le interazioni dei robot con l'ambiente circostante. Questa capacità permette ai robot di riconoscere oggetti, percepire texture, stimare la forza di presa e rispondere adeguatamente al contatto umano. Questi miglioramenti sono particolarmente rilevanti per applicazioni come la manipolazione di oggetti delicati, l'assistenza domestica e la collaborazione uomo-robot in ambienti industriali.[6]

Infine, i sensori tattili ad alta risoluzione aprono nuove possibilità nel riconoscimento di gesti. Sensori avanzati, come quelli basati su capacità mutua, consentono di rilevare e interpretare gesti complessi effettuati sulla superficie di un manipolatore, facilitando interazioni più sofisticate e intuitive tra uomo e robot.[4]

1.2.2 Assistenza Sanitaria

Un altro ambito di applicazione è il monitoraggio fisiologico. Attraverso l'uso di sensori indossabili dotati di componenti piezoresistivi o capacitivi, è possibile monitorare parametri vitali come la frequenza cardiaca, la pressione sanguigna e la respirazione. L'analisi dei segnali tattili provenienti dalla pelle consente di rilevare eventuali anomalie e di fornire feedback personalizzati e in tempo reale, migliorando così il controllo della salute del paziente.[7]

1.3 Limitazioni della tecnologia in commercio per costo e fabbricazione

Sebbene i sensori tattili siano promettenti per diverse applicazioni, come la robotica, l'assistenza sanitaria e l'interazione uomo-robot, la loro diffusione commerciale è ostacolata da alcune

limitazioni, tra cui i costi elevati.

La complessità di fabbricazione rappresenta una sfida significativa per molti sensori tattili, in particolare quelli ad alta risoluzione e flessibili, che richiedono processi complessi e costosi. Ad esempio, i sensori capacitivi necessitano spesso di materiali speciali sia per le piastre conduttive che per il dielettrico, mentre i sensori piezoelettrici devono essere realizzati con materiali puri e assemblati con tecniche specifiche [8].

Un'altra difficoltà riguarda la scalabilità della produzione, soprattutto per applicazioni come la pelle artificiale. Sebbene tecnologie emergenti, come la stampa 3D e la deposizione a getto d'inchiostro, offrano soluzioni promettenti per ridurre i costi di produzione, tali tecnologie necessitano ancora di perfezionamenti per garantire l'affidabilità e la durata dei sensori [9].

Inoltre, i sensori tattili richiedono l'integrazione di componenti elettronici sofisticati per l'elaborazione dei segnali, l'amplificazione e la conversione analogico-digitale. Questo aspetto, soprattutto nei sensori flessibili e indossabili, contribuisce ad aumentare notevolmente i costi complessivi di produzione[10].

La robustezza e la durata dei sensori tattili rappresentano un altro ostacolo: molti di essi sono ancora fragili e suscettibili a danni causati da usura, pressione eccessiva o condizioni ambientali avverse. Questo limita il loro impiego in contesti che richiedono alta affidabilità, come la robotica industriale [1].

Anche la risoluzione spaziale dei sensori tattili, cioè la loro capacità di distinguere tra due punti di contatto vicini, risulta spesso inferiore a quella della pelle umana. Questa limitazione riduce la loro capacità di percepire dettagli fini e texture complesse [9].

Infine, la mancanza di una standardizzazione rende difficile l'interoperabilità tra diversi dispositivi e piattaforme, ostacolando lo sviluppo di applicazioni e algoritmi universali [1].

1.4 Obiettivi della tesi

L'obiettivo principale di questo lavoro è l'identificazione dei pattern di pressione e l'anticipazione delle necessità dell'utente attraverso l'analisi dei segnali raccolti da sensori tattili a basso costo. In particolare, si intende sviluppare e validare algoritmi in grado di riconoscere movimenti specifici quali la volontà dell'utente di muoversi nelle quattro direzioni spaziali, utilizzando sensori che siano economicamente accessibili e facilmente integrabili in dispositivi indossabili o robotici. L'uso di sensori a basso costo è essenziale per rendere la tecnologia accessibile a un pubblico più ampio e promuovere la diffusione delle pelli sensoriali artificiali in applicazioni quotidiane come l'assistenza sanitaria o l'interazione uomo-robot. Inoltre, si cercherà di superare le limitazioni di precisione e affidabilità tipicamente associate ai sensori economici,

attraverso tecniche di elaborazione e classificazione dei segnali, migliorando così la capacità dei sistemi di riconoscere e adattarsi in tempo reale alle esigenze dell'utente.

Capitolo 2

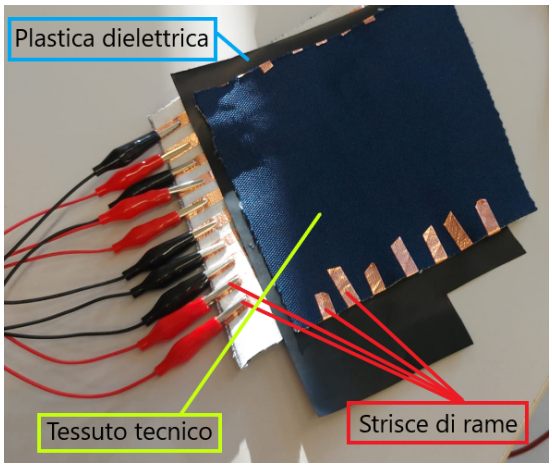
Materiali e Metodi

Per questo studio è stato selezionato un sensore di pressione piezoresistivo, principalmente per il suo conveniente rapporto "qualità-prezzo". Tale dispositivo sfrutta la variazione della resistenza elettrica di un materiale che avviene in risposta a uno stress meccanico, come l'applicazione di una forza pressoria. La struttura del sensore è costituita da strisce di rame disposte in modo perpendicolare l'una all'altra, collocate su lati opposti di un substrato plastico dielettrico. Questa configurazione consente di rilevare la variazione di pressione in punti alla distanza di circa 40mm ovvero la distanza media tra le strisce di rame di cui è composto. Il sensore è stato successivamente usato per l'addestramento di alcuni modelli classificatori per il riconscimento di cinque movimenti: avanti, indietro, rotazione a destra, rotazione a sinistra e fermo. L'obiettivo era quello di predire l'intenzione di movimento dell'utente che usava il deambulatre eventualmente per agevolarne la mobilità.

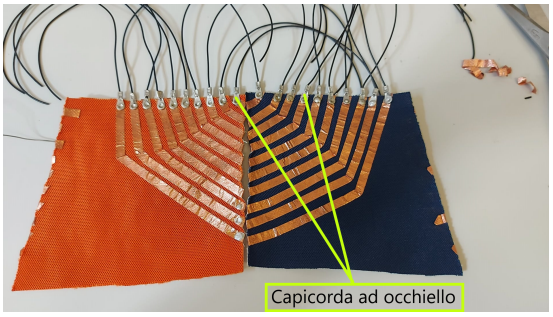
2.1 Selezione dei Materiali

La scelta dei materiali è stata cruciale per sviluppare un sensore economico ed efficace. Le strisce di rame adesive RS PRO sono state selezionate per la loro economicità e semplicità di utilizzo rispetto a sistemi di misurazione più complessi come l'EIT (Electrical Impedance Tomography). Queste strisce mostrate nelle figure 2.1a e 2.1b, disposte a reticolo, permettono di migliorare la precisione nella misurazione della resistenza della plastica dielettrica.

Per garantire che il sensore sia resistente e stabile, è stato scelto un tessuto tecnico in fibra di poliestere. Questo materiale offre durabilità e flessibilità, fondamentali per sopportare l'usura nel tempo, soprattutto quando è montato sulle manopole del deambulatore. La sua impermeabilità aggiunge un ulteriore livello di protezione, assicurando che il sensore possa funzionare in diverse condizioni ambientali senza compromettere le sue prestazioni.



(a) Materiali



(b) Capicorda ad occhiello

Figura 2.1: Materiali utilizzati per la costruzione del sensore

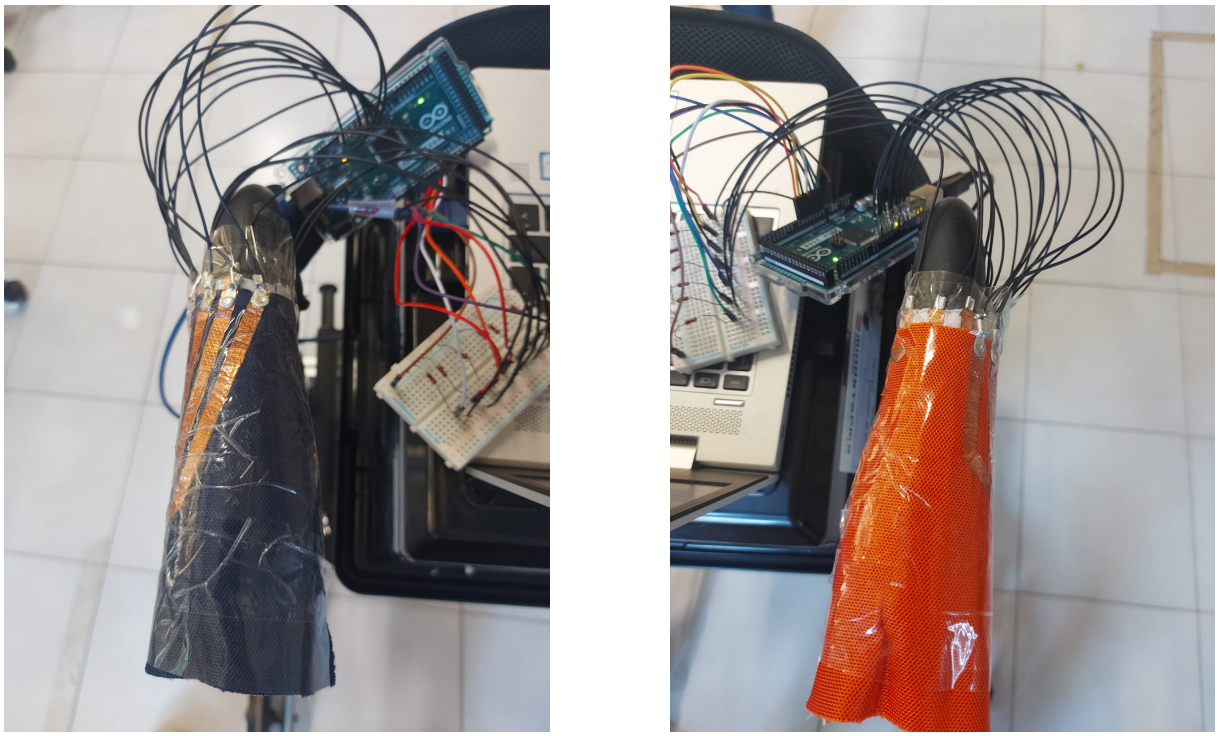
I resistori da 120 Ω e 220 Ω, con una tolleranza del ±5%, sono stati scelti per gestire il flusso di corrente nel circuito, garantendo la precisione della misurazione. Durante i test e la calibrazione del sistema, le pinze a coccodrillo si sono rivelate indispensabili per collegamenti temporanei ai punti di misura, mentre i capicorda a occhiello sono stati utilizzati per connessioni permanenti tra cavi e strisce di rame.

Si è scelto l'Arduino MEGA 2560 per la sua capacità di gestire un elevato numero di pin digitali e analogici. Questo lo rende ideale nel coordinare le connessioni tra le molteplici strisce di rame.[11][12]

2.2 Software per la raccolta dei dati

Per la raccolta dei dati, ho sviluppato un software in Python che utilizza la libreria Pygame per creare un'interfaccia grafica intuitiva mostrata nella figura 2.3, che guida l'utente durante il processo di acquisizione dei dati [13]. Il programma, una volta avviato, mostra all'utente una schermata con una serie di pulsanti, ognuno dei quali rappresenta uno dei cinque movimenti che devono essere registrati: avanti, indietro, sinistra, destra e fermo. L'utente seleziona un movimento cliccando su uno dei pulsanti, e a quel punto inizia un conto alla rovescia visuale sullo schermo che prepara l'utente all'inizio della registrazione.

Durante il conto alla rovescia mostrato nella figura 2.4, il programma consente all'utente di prepararsi per eseguire il movimento selezionato, che comporta l'applicazione di una pressione specifica su determinate aree e in momenti precisi. Questo processo permette la classificazione delle intenzioni dell'utente. Ad esempio, quando viene selezionato il movimento "avanti", rappresentato da una freccia verso l'alto visibile sullo schermo, il countdown inizia e, al suo



(a) Manopola sinistra (b) Manopola destra

Figura 2.2: Manopole sinistra e destra coi sensori montati

termine, i dati di pressione vengono raccolti per 7 secondi, durante i quali l'utente eseguirà il movimento scelto. Il peso corporeo e la direzione del passo generano un pattern di pressione caratteristico, utile per il riconoscimento del movimento. Per garantire maggiore accuratezza, è stato implementato un sistema di sincronizzazione tra i sensori delle manopole destra e sinistra. La registrazione avviene in modo alternato su uno dei due sensori, evitando la rilevazione simultanea su entrambi.

In figura 2.5 viene schematizzato il processo di acquisizione dei dati. I voltaggi del sensore vengono letti da un Arduino per righe e inseriti in una matrice 7x10, Una volta serializzati in JSON vengono inviati al computer dove la matrice viene inserita in un array in attesa della fine dell'acquisizione. Infine l'array di matrici viene serializzato e salvato in un file insieme al timestamp e al tipo di movimento associato. Lo schema semplifica il processo utilizzando un solo Arduino, ma in realtà i dati vengono acquisiti alternativamente da due Arduino, sincronizzando le letture tra di loro.

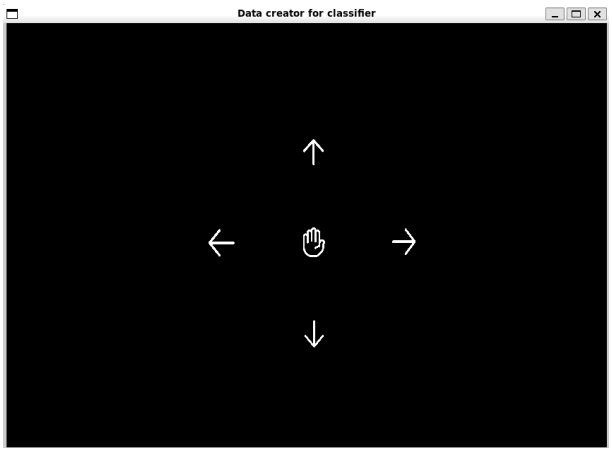


Figura 2.3: Main page gui

Figura 2.4: Count down gui

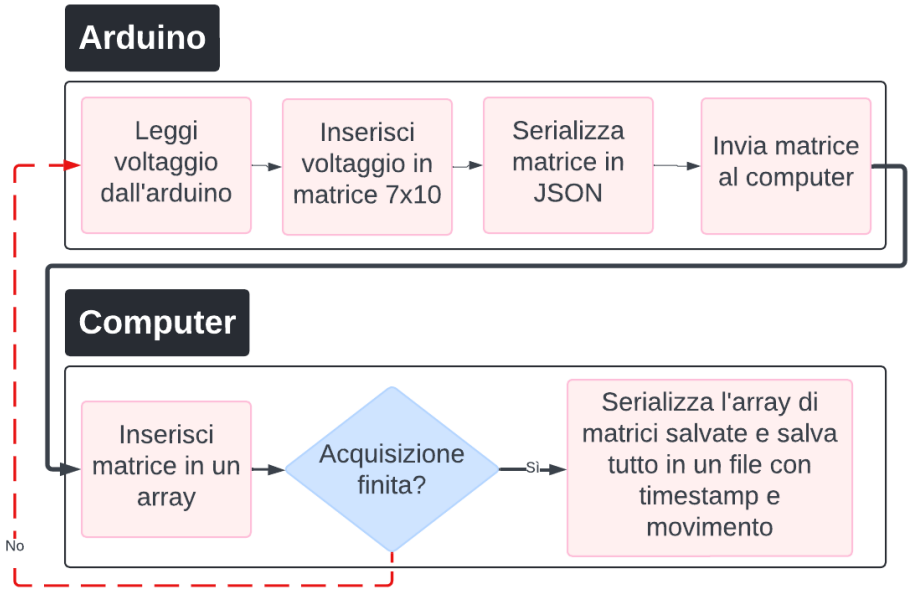


Figura 2.5: Schema del data processing

I file vengono posizionati in una struttura di directory organizzata per tipo di movimento, usando come nome una combinazione dell'azione eseguita e il timestamp corrente. Questo permette una facile gestione e accesso ai dati per la fase successiva di training del modello di classificazione. Di seguito è riportato un esempio di come i dati vengono salvati su file:

```
[
    // Dati del sensore di destra
    [
        [132,134,106,12,7,12,75],
        [38,93,102,17,56,16,34],
        [26,16,59,27,72,22,13],
        [18,52,61,22,75,19,14]
    ],
    // Dati del sensore di sinistra
    [
        [132,134,106,12,7,12,75],
        [38,93,102,17,56,16,34],
        [26,16,59,27,72,22,13],
        [18,52,61,22,75,19,14]
    ],
    // Dati del sensore di destra
    [
        [...],
         ...
    ],
    // Dati del sensore di sinistra
    [
         ...
    ]
    ...
]
```

Il formato JSON facilita l'organizzazione dei dati, rendendo possibile il loro utilizzo diretto in vari software di analisi dati e librerie di machine learning, senza bisogno di ulteriore preprocessing.

2.3 Software per la predizione del movimenti

Il software per la predizione dei movimenti è sviluppato in Python ed è progettato per addestrare e valutare diversi modelli di classificazione su un dataset. Di seguito è fornita una descrizione dettagliata delle sue funzionalità e delle dipendenze utilizzate.

2.3.1 Funzionalità del Software

Il programma si compone di diverse fasi:

1. **Caricamento dei Dati:** Il software inizia con il caricamento dei dati e delle etichette da una directory specificata. Questo viene effettuato tramite una funzione denominata `load_data_from_directory`. La directory di origine dei dati è specificata dalla variabile `DATA_DIRECTORY`.

2. **Preprocessing dei Dati:** Successivamente, i dati vengono pre-elaborati. In particolare, le caratteristiche vengono estratte e appiattite in vettori unidimensionali utilizzando `np.array(action).flatten()` [14]. Questo passaggio è fondamentale per preparare i dati all'addestramento dei modelli di classificazione in quanto sono precedentemente organizzati in forma matriciale mentre è necessario lavorare con degli array di numeri nella fase di addestramento.

3. **Divisione dei Dati:** I dati vengono suddivisi in set di addestramento e di test utilizzando la funzione `train_test_split` della libreria `scikit-learn` [15]. Il dataset viene diviso in due parti, con il 70% dei dati utilizzati per l'addestramento e il 30% per il test.

4. **Addestramento dei Modelli:** Il software addestra cinque modelli di classificazione differenti: Multinomial Naive Bayes, Support Vector Machines, K-Nearest Neighbors, Decision Tree Classifier e Random Forest. Questi classificatori verranno descritti nella sezione apposita "Metodi di classificazione".

5. **Valutazione dei Modelli:** Infine, viene caricato ciascun modello salvato e utilizzato il set di test per effettuare previsioni. L'accuratezza di ciascun modello viene calcolata confrontando le previsioni con le etichette reali. L'accuratezza è espressa come percentuale di previsioni corrette rispetto al numero totale di previsioni effettuate.

2.3.2 Dipendenze Utilizzate

Il software fa uso delle seguenti librerie e moduli:

- **NumPy (`numpy`):** Utilizzato per la manipolazione degli array e il flattening dei dati. La funzione `np.array(action).flatten()` serve a convertire ogni array di dati in un vettore unidimensionale.[14]

- **Scikit-learn (`sklearn`):** Una libreria fondamentale per il machine learning che offre vari strumenti per la modellazione.[15]

- **Joblib (`joblib`):** Utilizzato per salvare e caricare i modelli addestrati su disco. La funzione `joblib.dump` salva il modello, mentre `joblib.load` lo carica per ulteriori previsioni.[16]

2.4 Metodi di classificazione

I metodi di classificazione sono tecniche fondamentali nell'ambito dell'apprendimento automatico, utilizzati per assegnare etichette a nuovi dati basandosi su esempi già noti. Tra i più popolari vi sono il Random Forest, il K-Nearest Neighbors (KNN), il Support Vector Machines (SVM), il Decision Tree e il Multinomial Naive Bayes (MNB). Al fine di classificare le intenzioni di movimento a partire dai dati di pressione del sensore in figure 2.2b e 2.2a è stato effettuato un training di diversi modelli il cui funzionamento viene discusso in questa sezione.

2.4.1 Random Forest

Random Forest è un algoritmo di apprendimento automatico basato su alberi di decisione, utilizzato sia per problemi di classificazione che di regressione. È un tipo di algoritmo di ensemble che combina la previsione di diversi alberi decisionali [17].

Questo algoritmo funziona creando una "foresta" di alberi decisionali, ognuno dei quali è addestrato su un sottoinsieme casuale dei dati di addestramento. I risultati di tutti gli alberi vengono quindi combinati (ad esempio, attraverso una media o un voto di maggioranza) per ottenere una previsione più accurata e stabile rispetto a un singolo albero decisionale [17].

La fase di addestramento, volta a popolare la "foresta" si divide in varie fasi:

- **Campionamento Casuale con Sostituzione (Bagging):** Random Forest utilizza una tecnica chiamata *Bagging* (Bootstrap Aggregating). Per costruire ogni albero decisionale, viene estratto un campione casuale dal set di dati originale con sostituzione, cioè alcuni punti dati possono essere selezionati più volte. Questo porta a una varietà tra gli alberi [17].

- **Selezione Casuale delle Variabili:** Durante la costruzione di ogni albero, a ogni nodo viene associato un sottoinsieme casuale di caratteristiche (feature) piuttosto che l'intero set, per decorrelare gli alberi [17].

- **Costruzione degli Alberi:** Ogni albero decisionale è costruito in modo tradizionale, dividendo i nodi in base alla caratteristica che meglio separa i dati in quel nodo. Gli alberi vengono addestrati completamente fino alle foglie [17].

Per un problema di classificazione, la previsione finale è determinata dalla classe che riceve la maggioranza dei voti da tutti gli alberi [17]. Questo processo di voto di maggioranza garantisce che la decisione finale non sia influenzata eccessivamente da un singolo albero, il quale potrebbe avere errori o bias. Ogni albero fornisce un contributo alla previsione, e aggregando le risposte si ottiene una soluzione più equilibrata e rappresentativa del dataset. In particolare, questa metodologia riduce la varianza complessiva del modello, permettendo a Random Forest di generalizzare meglio su nuovi dati rispetto a singoli alberi decisionali. Inoltre, nel caso di problemi di regressione, il processo di aggregazione può avvenire tramite la media delle previsioni di tutti gli alberi, fornendo una stima continua che considera l'output di ciascun albero in modo equo.[17]

2.4.2 Multinomial Naive Bayes (MNB)

Il Multinomial Naive Bayes (MNB) è un classificatore probabilistico che assume che le caratteristiche siano condizionatamente indipendenti, dato il valore della classe di appartenenza. Questa assunzione, nota come "naive" (ingenua), semplifica notevolmente il processo di classificazione, rendendo MNB un algoritmo computazionalmente efficiente e facile da implementare. È particolarmente adatto per dati discreti, come nei problemi di classificazione di testi, dove le caratteristiche sono rappresentate dalla frequenza di parole nei documenti.

Il termine "multinomiale" si riferisce a un tipo di distribuzione probabilistica che generalizza la distribuzione binomiale ai casi in cui ogni osservazione può appartenere a più di due classi. In un modello multinomiale, il risultato di un esperimento non è limitato a due esiti, ma può appartenere a k possibili categorie. Questa caratteristica rende il MNB particolarmente adatto per dati discreti, come conteggi o frequenze, tipici nei documenti di testo.

Il funzionamento di MNB si basa sull'idea che ogni caratteristica del dato (ad esempio, una parola in un testo o una variabile numerica in un dataset) contribuisca in modo indipendente alla probabilità che quel dato appartenga a una specifica classe. L'algoritmo calcola la probabilità che una caratteristica si presenti in ciascuna classe, partendo da un insieme di dati già etichettati, e somma queste probabilità per decidere la classe più probabile per un nuovo dato. Anche

se l'assunzione di indipendenza è spesso irrealistica, l'approccio semplificato consente all'algoritmo di essere molto efficiente e di ottenere comunque buoni risultati in molte applicazioni pratiche [18].

2.4.3 Support Vector Machines (SVM)

Il Support Vector Machines (SVM) è un metodo di classificazione che cerca di trovare un iperpiano che separa le classi con il margine più ampio. Gli SVM sono efficaci in spazi ad alta dimensione e sono robusti ai dati rumorosi. Sebbene gli SVM abbiano un buon potenziale di generalizzazione, possono essere sensibili ai parametri di configurazione e ai dati non bilanciati. [19] Di base, l'SVM è pensato per separare due classi, quindi quando ci sono più di due categorie, si applicano strategie specifiche per estendere il suo funzionamento. Le due tecniche più comuni sono:

- **One-vs-One (OvO):** In questa strategia, l'SVM crea un classificatore per ogni possibile coppia di classi. Se ci sono n classi, l'algoritmo costruisce un classificatore per ciascuna delle $\frac{n(n-1)}{2}$ combinazioni di classi.

  Esempio: Supponiamo di avere tre classi, A, B e C. In questo caso, il metodo OvO costruisce un classificatore per distinguere la classe A dalla classe B, un altro per distinguere la classe A dalla classe C, e infine un classificatore per separare la classe B dalla classe C. Per classificare un nuovo punto, ciascun classificatore OvO fa una predizione, e la classe che vince più confronti (voti) viene scelta.

  Tra i vantaggi del metodo OvO, vi è il fatto che funziona particolarmente bene quando il numero di classi è ridotto. Inoltre, richiede generalmente meno memoria, poiché ciascun classificatore si occupa di distinguere solo una coppia di classi alla volta, gestendo quindi solo una parte limitata del problema complessivo. Tuttavia, uno svantaggio significativo di questo approccio è che, all'aumentare del numero di classi, il numero di classificatori necessari cresce rapidamente, rendendo il calcolo sempre più dispendioso in termini di tempo.[20]

- **One-vs-Rest (OvR):** Il metodo OvR funziona costruendo un classificatore separato per ciascuna classe. In questo approccio, ogni classe viene considerata come "positiva" (indicata con 1), mentre tutte le altre classi vengono trattate come "negative" (indicata con 0). Di conseguenza, se ci sono N classi, saranno necessari N classificatori.

  Esempio: Se consideriamo tre classi, A, B e C, il metodo OvR costruisce un modello per distinguere A dalle altre due (B e C), un altro per distinguere B da A e C, e infine uno per separare C dalle altre due. Per classificare un nuovo punto, vengono utilizzati tutti i

modelli creati con OvR. Ognuno di essi restituisce una predizione, e alla fine viene scelta la classe che ha ottenuto la probabilità più alta.

Tra i principali vantaggi di questo approccio, vi è la sua semplicità e velocità di implementazione rispetto al metodo OvO. Inoltre, il numero di classificatori richiesti cresce in modo lineare con il numero di classi, poiché per ogni classe ne richiede solo uno. Tuttavia, lo svantaggio principale è che, se le classi non sono ben bilanciate, alcuni classificatori potrebbero incontrare difficoltà nel separarle, riducendo l'efficacia complessiva del modello.[20]

2.4.4 K-Nearest Neighbors (KNN)

Il K-Nearest Neighbors è un metodo di classificazione basato sulla distanza tra punti che valuta un campione in base alla maggioranza dei suoi vicini più prossimi. Tra i vantaggi, spiccano la sua semplicità e facilità di implementazione, poiché non richiede un ampio addestramento e si basa direttamente sui dati. È anche flessibile e efficace in situazioni multi-classe. In contesti con dati ad alta dimensione risulta computazionalmente costoso specialmente su set di dati grandi. All'aumentare delle dimensioni si manifesta anche un problema noto come "curse of dimensionality" che rende sempre più difficile la classificazione. È sensibile al rumore dei dati in quanto anche un singolo dato etichettato erroneamente potrebbe compromettere il sistema di classificazione finale. [21]

Quando si deve classificare un nuovo dato, KNN calcola la distanza tra il punto da classificare e tutti gli altri punti nel dataset. Le distanze possono essere calcolate utilizzando varie metriche, ma la più comune è la distanza euclidea. Vengono quindi eseguiti diversi step per arrivare alla classificazione del punto di test:

- **Dati di Addestramento:** Iniziamo con un insieme di dati di addestramento, che contiene diversi esempi con le loro caratteristiche (features) e le relative etichette (classi). Questi dati serviranno come riferimento per la classificazione.[21]

- **Scelta del Valore di K:** Il primo passo pratico è scegliere un valore per K, che rappresenta il numero di vicini che considereremo. Ad esempio, se scegliamo $K = 3$, il modello analizzerà i 3 punti più vicini al nuovo punto che vogliamo classificare.[21]

- **Misurazione della Distanza:** Per identificare i vicini, dobbiamo calcolare la distanza tra il nuovo punto e tutti i punti nel nostro insieme di dati. Le distanze più comuni utilizzate

sono la distanza euclidea, definita come:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

e la distanza di Manhattan, calcolata come:

$$d(x, y) = \sum_{i=1}^{n}|x_i - y_i|$$

[21]

- **Trova i Vicini:** Dopo aver calcolato le distanze, il modello identifica i K punti più vicini al nuovo punto in base a queste distanze.[21]
- **Votazione:** Infine, per classificare il nuovo punto, il modello guarda le etichette delle classi dei K vicini. Ogni vicino ”vota” per la propria classe, e il nuovo punto viene classificato nella classe che ha ricevuto il maggior numero di voti.[21]

2.4.5 Decision Tree Classifier

Il Decision Tree è un modello di apprendimento automatico utilizzato per la classificazione e la regressione, che rappresenta le decisioni in forma di un albero. Ogni nodo interno dell'albero corrisponde a una caratteristica del dataset, mentre le ramificazioni rappresentano i risultati di una domanda su quella caratteristica. Il processo di ramificazione continua fino a raggiungere un criterio di arresto, come un numero massimo di livelli o la creazione di nodi puri. Al termine, ogni foglia dell'albero rappresenta una classe (nel caso della classificazione) o un valore (nel caso della regressione), consentendo così di fare previsioni sui nuovi dati. [22]

Al contrario di modelli di classificazione come KNN dove la fase di addestramento è di facile implementazione, in un decision tree si verifica il processo opposto. La fase di addestramento è di fondamentale importanza in quanto la capacità di classificazione del modello è intrinseca alla forma che l'albero assume. Il processo di creazione del Decision Tree avviene in diverse fasi:

- **Selezione delle caratteristiche:** Il processo inizia con il set di dati completo. L'albero cerca di trovare la migliore caratteristica su cui suddividere i dati. Questo viene fatto utilizzando misure di impurità come il *Gini Index* o l'*Entropia*, che valutano quanto i dati sono ”puri” dopo la suddivisione. [22]
- **Creazione dei nodi:** Una volta scelta la caratteristica migliore, i dati vengono divisi in base alla stessa. Ogni sottogruppo diventa un nodo figlio. Questo processo si ripete per

ciascun nodo, continuando a suddividere i dati fino a quando non si raggiunge un criterio di arresto, come un numero massimo di livelli dell'albero o quando i nodi diventano omogenei (ossia contengono solo una classe). [22]

Quando si presenta un nuovo punto, il classificatore segue il percorso nell'albero, rispondendo alle domande ai nodi fino a raggiungere una foglia. La classe associata a quella foglia è quella assegnata al nuovo punto. [22]

Capitolo 3

Risultati

3.1 Reazione del sensore alla pressione

Il sensore, collegato ad un Arduino, permette di ottenere i valori del voltaggio della corrente passante per ciascuno dei nodi della matrice su cui si sviluppa. Ecco una rappresentazione grafica di un test eseguito per verificarne la funzionalità. Si riporta di seguito l'insieme dei valori letti dal sensore in un intervallo di tempo che vengono graficamente disposti ciascuno nel proprio punto di lettura. Si schematizza per semplicità il sensore come una griglia 7x10 nella tabella 3.1

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	1	4	85	0	0	0
0	2	4	326	0	0	0
0	0	0	18	0	0	0
14	21	0	139	1	0	1
0	3	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Tabella 3.1: Esempio di dati restituiti dalla porta seriale con il prototipo

La griglia è stata ricavata dai dati trasmessi dalla porta seriale. Per semplicità viene usato lo standard JSON. Si riporta di seguito quanto trasmesso dal sensore:

```
[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,1,4,85,0,0,0],[0,2,4,326,0,0,0],
 [0,0,0,18,0,0,0],[14,21,0,139,1,0,1],[0,3,0,0,0,0,0],[0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0],[0,0,0,0,0,0,0]]
```

3.2 Addestramento dei modelli sui movimenti campione

Dopo aver addestrato tutti i modelli sullo stesso training set e averli valutati sul medesimo testing set, i risultati mostrano una variazione nelle prestazioni dei diversi algoritmi. Le percentuali riportate sono state calcolate come il rapporto numero di classificazioni corrette e il numero di classificazioni totali in percentuale. Il modello *Multinomial Naive Bayes* (MNB) ha raggiunto un'accuratezza dell'84,21%, superando sia il modello *Support Vector Machine* (SVM) con l'82% di accuratezza, sia *K-Nearest Neighbors* (KNN) che ha ottenuto l'80%. Il modello *Random Forest* (RF) ha mostrato la sua superiorità con un'accuratezza dell'86,84%, risultando il più efficace nel classificare i movimenti. Il modello *Decision Tree* (DT), invece, ha riportato un'accuratezza del 76,32%. Nella figura 3.1 è presente un istogramma che riassume quanto detto.

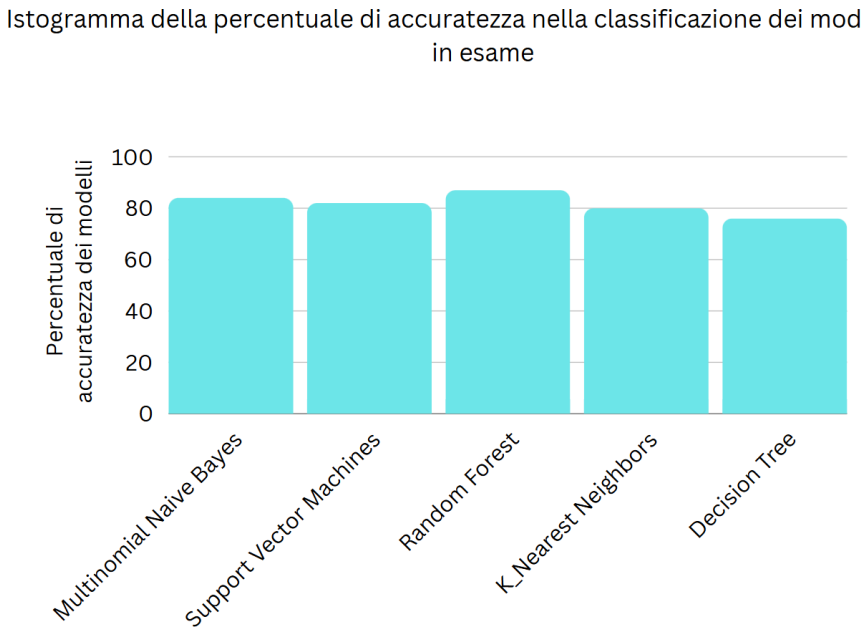


Figura 3.1: Schema del data processing

3.3 Visualizzazione dell'addestramento del Random Forest

Nelle figure 3.2, 3.3 e 3.4 viene data una rappresentazione grafica dei pesi delle feature del modello Random Forest dopo l'addestramento. A tale scopo il vettore delle importanze delle features è riportato in forma matriciale così da rendere visualizzabile, per ciascun frame raccolto dal sensore, il cambiamento dei valori delle importanze delle features. I dati di pressione infatti non vengono valutati dal modello istante per istante, ma vengono classificati i dati raccolti in un arco temporale definito nel programma di acquisizione. Questo rende necessario raccogliere per

ogni istante tutta la matrice dei punti di rilevazione del sensore portando ad una rappresentazione come figura. Il modello quindi, come si vede dall'immagine, valuta anche i cambiamenti di pressione nel tempo. Per ciascuna delle 50 matrici vengono evidenziate le feature che presentano un peso non nullo usando la scala colorata a destra (ogni colore è associato ad un numero che è l'importanza delle feature). Il numero di matrici è stato scelto come il più piccolo campione preso durante la fase di raccolta dei dati. Sette secondi di dati corrispondo circa a 50/60 matrici di dati del sensore (il valore del tempo di acquisizione è discusso nel capitolo "Discussione dei risultati") . Siccome è necessario che tutti i samples per l'addestramento abbiano la stessa dimensione i dati in eccesso sono stati rimossi. Le figure 3.2, 3.3 e 3.4 sono state divise per semplificarne la lettura in quanto occupavano troppo spazio.

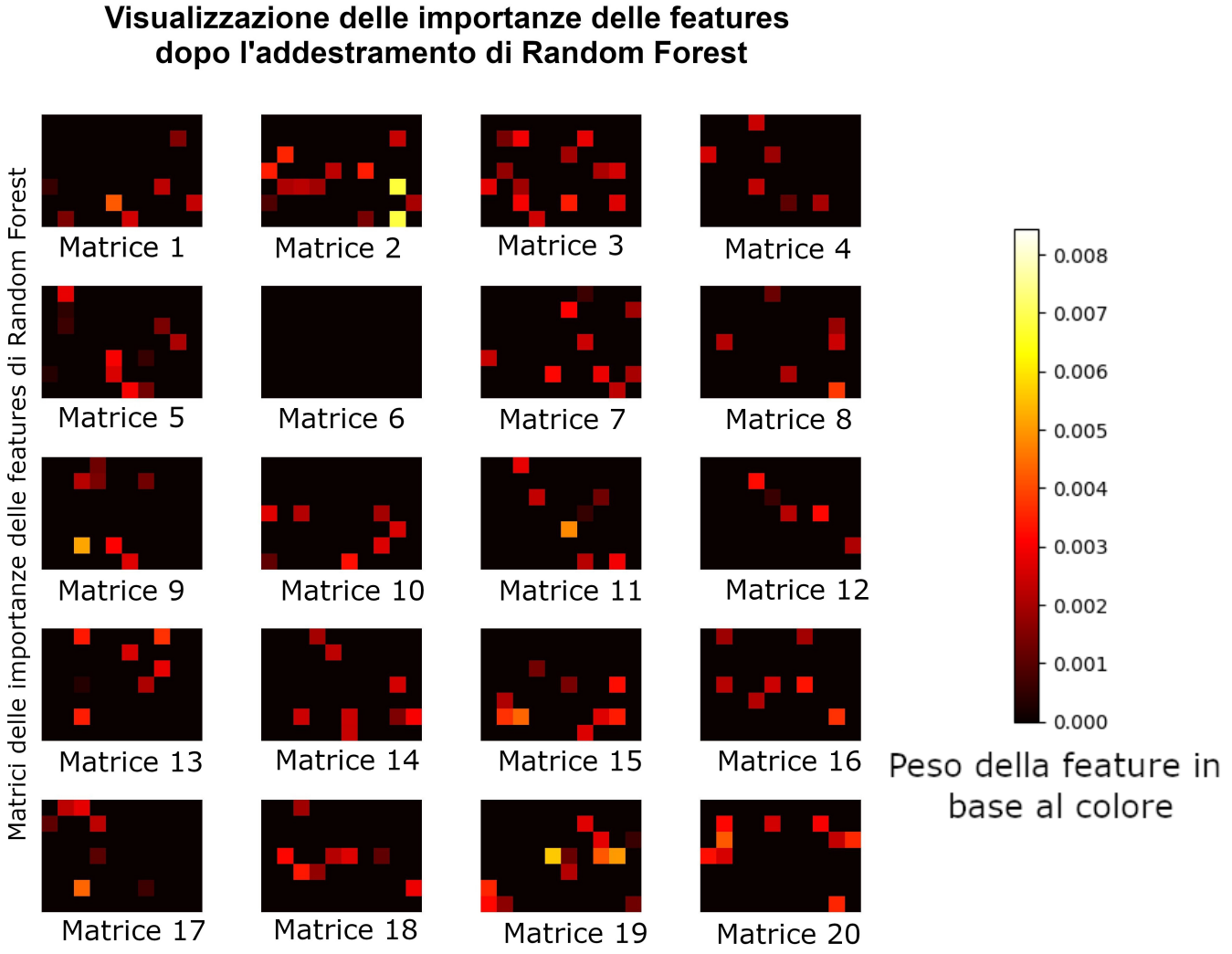


Figura 3.2: Visualizzazione dei pesi delle feature del Random Forest dopo l'addestramento. A destra, la scala dei colori rappresenta i pesi.

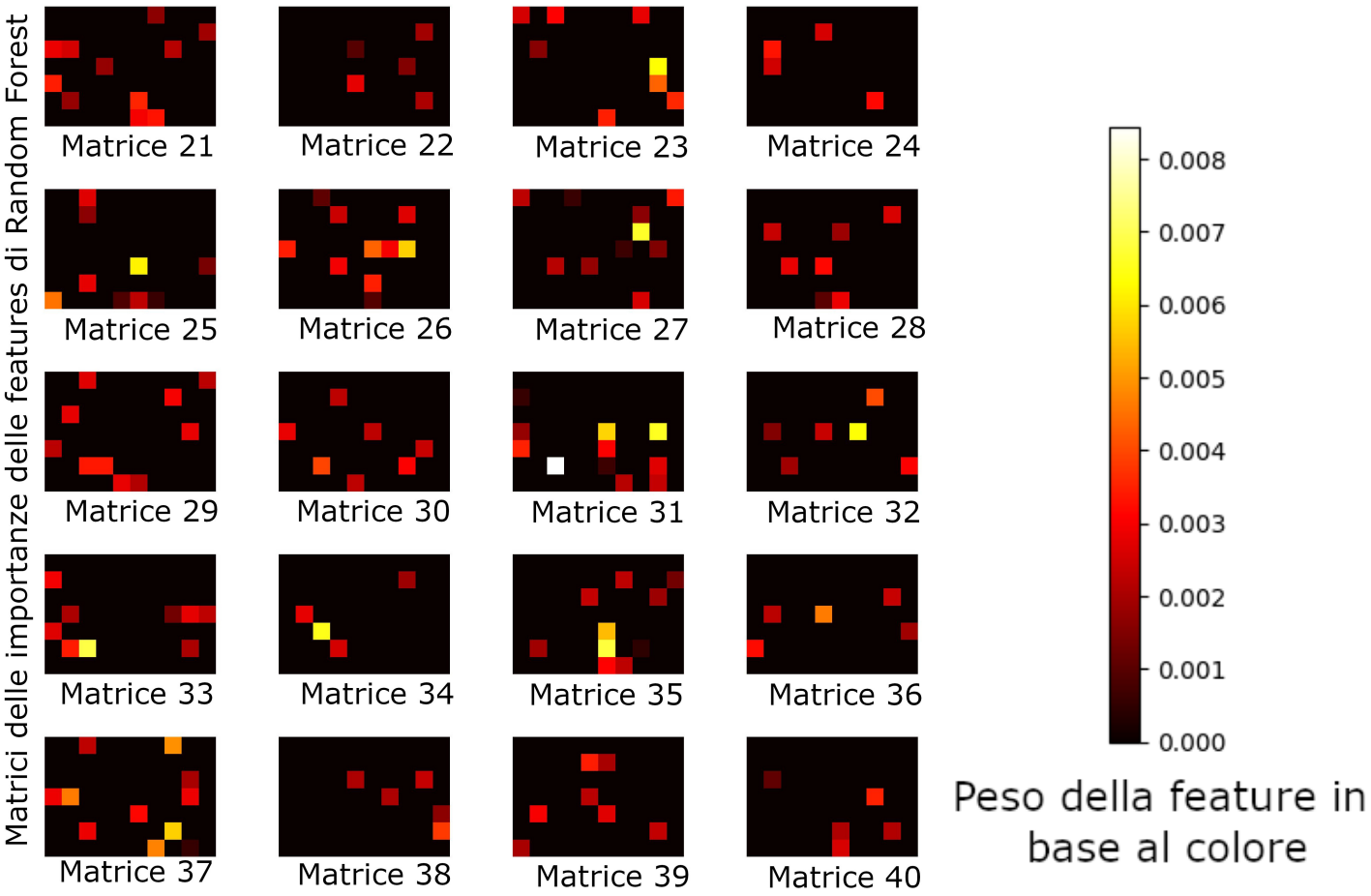


Figura 3.3: Visualizzazione dei pesi delle feature del Random Forest dopo l'addestramento. A destra, la scala dei colori rappresenta i pesi.

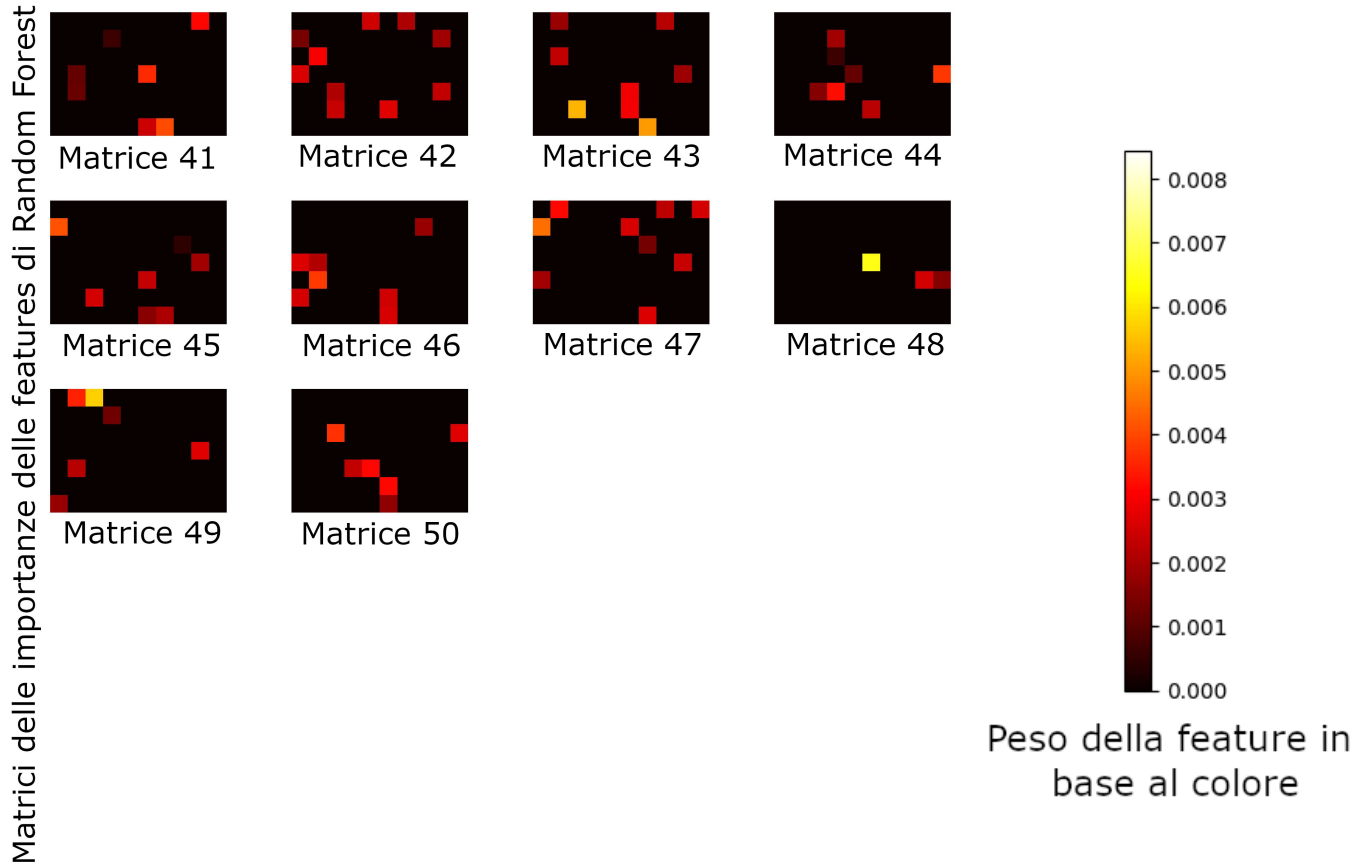


Figura 3.4: Visualizzazione dei pesi delle feature del Random Forest dopo l'addestramento. A destra, la scala dei colori rappresenta i pesi.

3.4 Comparazione di Random Forest con gli altri modelli analizzati

Random Forest si è dimostrato più efficace rispetto agli altri modelli classificatori utilizzati per diversi motivi. Prima di tutto, la sua robustezza all'overfitting è un vantaggio significativo. A differenza di modelli come Decision Tree, che tendono a sovradattare i dati di addestramento creando alberi troppo specifici, Random Forest utilizza la tecnica del bootstrap aggregating prevenendo questa problematica.[22]

In secondo luogo, Random Forest gestisce molto bene i dati complessi e non lineari. Gli alberi decisionali multipli possono modellare queste relazioni complesse grazie alla loro struttura gerarchica basata su domande binarie. Combinando le decisioni di molti alberi, il modello può rappresentare in modo più accurato le interazioni tra le variabili, migliorando così le prestazioni complessive.[17]

Un altro vantaggio chiave è la sua robustezza ai dati mancanti e rumorosi. Poiché l'algoritmo si basa su una media o un voto di maggioranza tra gli alberi, errori o dati mancanti in un singolo albero hanno un impatto limitato sul risultato finale. Questo è particolarmente utile nel contesto dei sensori di pressione, dove le imperfezioni nella raccolta dei dati sono comuni. Al contrario, modelli come SVM o KNN possono essere più sensibili a tali anomalie nei dati.[17]

Infine, Random Forest offre la possibilità di valutare l'importanza delle caratteristiche (feature importance), fornendo una comprensione chiara di quali variabili influenzano maggiormente le previsioni. Questo è utile per ottimizzare ulteriormente il modello e per interpretare meglio il comportamento dei dati di pressione. Al contrario, modelli come SVM e KNN mancano di questa capacità di interpretazione.[17][21]

Capitolo 4

Discussione dei risultati

Il sensore ha dimostrato di rilevare correttamente la pressione applicata, garantendo dati affidabili per l'analisi. Inoltre, l'uso del formato JSON ha semplificato in modo efficace il trasferimento dei dati a livello software, rendendo la comunicazione tra l'Arduino e il computer più agevole. I modelli utilizzati riescono a classificare, seppure con precisioni differenti, i movimenti dell'utente, confermando la validità dell'approccio adottato. Tuttavia, un limite significativo emerso riguarda la velocità di classificazione: attualmente, sono necessari 7 secondi di dati per ottenere una sufficiente accuratezza nell'identificazione delle intenzioni dell'utente. Questo tempo di risposta risulta ancora troppo lento per garantire un'assistenza proattiva in tempo reale, limitando potenzialmente l'applicabilità del sistema in scenari pratici, dove una risposta immediata è cruciale.

La lunghezza del periodo di acquisizione dei dati è stata determinata in modo arbitrario a causa del tempo limitato a disposizione per lo sviluppo e i test del sistema. Sebbene questo abbia permesso di ottenere risultati preliminari sufficienti per valutare il funzionamento del modello, non si è trattato di una scelta ottimizzata. In futuro, sarà essenziale migliorare il sistema lavorando sulla riduzione progressiva della durata dell'acquisizione, valutando via via la quantità minima di dati necessaria per mantenere un livello adeguato di accuratezza nella classificazione. Questo processo di ottimizzazione dovrebbe prevedere test ripetuti per identificare il rapporto ideale tra la lunghezza dei dati raccolti e l'affidabilità dei risultati prodotti dai modelli. Riducendo il tempo necessario per la raccolta dei dati senza compromettere l'accuratezza delle predizioni, si potrebbe migliorare significativamente la rapidità del sistema, rendendolo più efficace per un'assistenza proattiva e immediata, soprattutto in contesti in cui le decisioni devono essere prese in tempo reale.

Sono stati eseguiti test unicamente su soggetti sani che simulavano problemi di mobilità, il che potrebbe non riflettere completamente le condizioni reali di utilizzo. Un fattore critico che potrebbe influenzare significativamente le predizioni del modello è la distribuzione del

peso corporeo in casi concreti. Ad esempio, una persona con dolore a una gamba potrebbe spostare il peso sull'altra per compensare, e il modello potrebbe interpretare erroneamente questo comportamento come un tentativo di girare in una direzione. Questo evidenzia una limitazione nella capacità del sistema di generalizzare in contesti reali, rendendo necessarie ulteriori prove su utenti con condizioni motorie specifiche per migliorare l'accuratezza delle predizioni e garantire una corretta interpretazione delle intenzioni dell'utente in situazioni cliniche reali. A tal proposito potrebbe essere necessario sviluppare un sistema che permetta di addestrare rapidamente i modelli in base alle caratteristiche specifiche dell'utilizzatore, personalizzando così le predizioni e migliorando l'efficienza dell'assistenza.

Un altro aspetto da considerare è la forma delle maniglie del deambulatore, che potrebbe influenzare il modo in cui l'utente applica la pressione e, di conseguenza, la raccolta dei dati. Un design ergonomico e ottimizzato potrebbe migliorare ulteriormente l'accuratezza delle rilevazioni e facilitare l'uso del sistema per un'ampia gamma di utenti. Nel nostro caso, non è stato possibile effettuare uno studio approfondito sull'anatomia delle mani e sull'impugnatura del dispositivo, se non su soggetti sani. Questo ha limitato la possibilità di ottimizzare la distribuzione dei punti di rilevamento della pressione in modo specifico per utenti con diverse condizioni motorie o fisiche. Un'analisi più dettagliata della disposizione delle mani sul deambulatore, soprattutto in persone con difficoltà motorie o patologie, potrebbe rivelare la necessità di una diversa e più mirata collocazione dei punti di rilevamento. Disporre i sensori in aree dove è più probabile che le mani applichino pressione, ad esempio sui palmi o sulle dita, migliorerebbe la precisione nella predizione dei movimenti e ridurrebbe il rischio di interpretazioni errate del comportamento dell'utente.

In sintesi si conferma la validità dell'approccio tuttavia, le migliorie proposte, come l'ottimizzazione della lunghezza del periodo di acquisizione, una disposizione più accurata dei punti di rilevazione in base allo studio anatomico delle mani e l'addestramento personalizzato per ciascun utente, renderebbero il progetto ancora più efficace e pronto per un utilizzo in ambito industriale. Nonostante le attuali limitazioni, in particolare per quanto riguarda la velocità di risposta del sistema, questo rimane comunque in grado di classificare correttamente i movimenti basandosi sui pattern di pressione, risultando utile in contesti dove la rapidità nell'interpretare gli input non è cruciale.

Bibliografia

[1] D. Silvera-Tawil, D. Rye e M. Velonaki, «Artificial skin and tactile sensing for socially interactive robots: A review,» *Robotics and Autonomous Systems*, vol. 63, pp. 230–243, 2015, Advances in Tactile Sensing and Touch-based Human Robot Interaction, issn: 0921-8890. doi: https://doi.org/10.1016/j.robot.2014.09.008. indirizzo: https://www.sciencedirect.com/science/article/pii/S0921889014001833.

[2] M. Jamshidi, C. B. Park e F. Azhari, «An EIT-based piezoresistive sensing skin with a lattice structure,» *Materials Design*, vol. 233, p. 112 227, 2023, issn: 0264-1275. doi: https://doi.org/10.1016/j.matdes.2023.112227. indirizzo: https://www.sciencedirect.com/science/article/pii/S0264127523006421.

[3] R. Dahiya e G. Metta, «Synthetic and Bio-Artificial Tactile Sensing A Review,» *Sensors*, vol. 13, n. 6, 2013.

[4] J. Sohrabi, M. Görner, M. Hoffmann e T. Asfour, «Human-Like Artificial Skin Sensor for Physical Human-Robot Interaction,» *IEEE Sensors Journal*, vol. 21, n. 18, 2021.

[5] K. Liu, Y. Wu, S. Wang et al., «Artificial sensitive skin for robotics based on electrical impedance tomography,» *Advanced Intelligent Systems*, vol. 2, n. 5, 2020.

[6] C. Lucarotti, C. M. Oddo, N. Vitiello e M. C. Carrozza, «Synthetic and Bio-Artificial Tactile Sensing: A Review,» *Sensors*, vol. 13, n. 2, pp. 1435–1466, 2013, issn: 1424-8220. doi: 10.3390/s130201435. indirizzo: https://www.mdpi.com/1424-8220/13/2/1435.

[7] P. Claver, N. Charlon, J. Descamps-Mandine, G. Coquerel e E. Foltête, «Recent progress in flexible pressure sensors based electronic skin,» *Advanced Engineering Materials*, vol. 23, n. 1, 2021.

[8] J. Ulmen e M. Cutkosky, «A robust, low-cost and low-noise artificial skin for human-friendly robots,» in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 4836–4841. doi: 10.1109/ROBOT.2010.5509295.

[9] N. Koutchoukali, C. Ramakrishnan e M. Sawan, «Stretchable, Flexible, Scalable Smart Skin Sensors for Measurement of Shearing Forces on a Surface,» *Sensors*, vol. 18, n. 4, 2018.

[10] H. Alirezaei, A. Nagakubo e Y. Kuniyoshi, «A tactile distribution sensor which enables stable measurement under high and dynamic stretch,» in *2009 IEEE Symposium on 3D User Interfaces*, 2009, pp. 87–93. doi: 10.1109/3DUI.2009.4811210.

[11] Arduino. (2024), indirizzo: https://www.arduino.cc/reference/it/language/functions/analog-io/analogread/.

[12] Arduino. (2024), indirizzo: https://www.arduino.cc/reference/en/language/variables/constants/highlow/.

[13] P. Community, *Pygame: Python Game Development*, Accessed: 2024-09-19, 2023. indirizzo: https://www.pygame.org/docs/.

[14] NumPy Developers, *NumPy Documentation*, 2023. indirizzo: https://numpy.org/doc/stable/.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort et al., *Scikit-learn: Machine Learning in Python*, 2011. indirizzo: https://scikit-learn.org/stable/documentation.html.

[16] J. Developers, *Joblib: Lightweight Pipelines for Python*, Accessed: 2024-09-19, 2023. indirizzo: https://joblib.readthedocs.io/en/stable/.

[17] L. Breiman, «Random forests,» *Machine learning*, vol. 45, n. 1, 2001.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[19] C. Cortes e V. Vapnik, «Support-vector networks,» *Machine learning*, vol. 20, n. 3, 1995.

[20] A. C. Lorena, A. C. P. L. F. Carvalho e J. M. Gama, «A review on the combination of binary classifiers in multiclass problems,» *Artificial Intelligence Review*, vol. 30, n. 1-4, pp. 19–37, 2008.

[21] T. Cover e P. Hart, «Nearest neighbor pattern classification,» *IEEE transactions on information theory*, vol. 13, n. 1, 1967.

[22] J. R. Quinlan, «Induction of decision trees,» *Machine learning*, vol. 1, n. 1, 1986.