# ANALISI DEI DATI SENSORIALI CON THINGY52: MONITORAGGIO E RISULTATI

## • Introduzione

Durante il corso, ci è stato fornito un dispositivo Thingy52. Si tratta di un apparecchio hardware sviluppato da Nordic Semiconductor, progettato per le applicazioni IoT (Internet of Things). Al suo interno si trovano diversi sensori e ne sono stati analizzati 3:

- Accelerometro: misura l'accelerazione lungo i tre assi (x, y, z).
- Giroscopio: misura la velocità angolare, ovvero la rotazione attorno ai tre assi.
- Bussola: rileva l'orientamento del dispositivo rispetto al campo magnetico terrestre, misurando il campo magnetico lungo i tre assi. (nel progetto non è stata utilizzata)

Il Thingy52 si connette a un computer o smartphone tramite Bluetooth Low Energy (BLE), consentendo la raccolta di dati in tempo reale da questi sensori.

## • Il software

Nella parte iniziale del corso ci sono state fornite delle basi del linguaggio di programmazione Python. In particolare, uno degli aspetti fondamentali analizzati sono state le classi.

Una classe è un modello strutturato per creare oggetti, che sono istanze di quella classe. Ogni classe può contenere:

- Attributi: variabili che descrivono le caratteristiche di un oggetto.
- Metodi: funzioni che descrivono il comportamento di un oggetto.

Nel progetto è stata creata una classe chiamata Thingy52Client per gestire la connessione al dispositivo Thingy52 e raccogliere i dati dai sensori.

**IMPLEMENTAZIONE DEL CODICE**

È stato implementato programma Python per raccogliere i dati di tre azioni a scelta dal Thingy52. Le azioni scelte sono state “mescolare” e “tagliare” e “stare_fermo”.

- STEP 1: RACCOLTA E SALVATAGGIO DEI DATI

Attraverso l’invio dello script main_class è stato possibile scansionare i dispositivi BLE nelle vicinanze, individuare tra la lista dei dispositivi trovati l’apparecchio Thingy52 e connettersi a esso. Una volta terminato il collegamento al dispositivo, sono stati registrati e salvati nei file .csv i dati relativi all’accelerometro e al giroscopio delle azioni scelte. Il nome del file è scelto dall’utente tramite input, in questo caso si faceva riferimento a “mescolare” , “tagliare” e “stare_fermo”.

- STEP 2: TRAINING DEI DATI

Nella seconda parte, è stato inviato da terminale la parte di codice relativa al train (addestramento del modello). In particolare, il dataset che costituisce tutti i dati raccolti in precedenza viene suddiviso in più parti (numero di k-folds) e in ogni ciclo si scelgono delle parti che funzionano da test set e altre per il training test e il validation test:

- il training set costituisce la parte più grande e serve per istruire il modello
- Il validation test serve per accertarsi che il modello stia imparando nel modo corretto e migliorarlo alla fine di ogni ciclo di training
- Il test set è lo step finale ed utile per capire quanto è efficiente il modello

Questo processo viene ripetuto k volte in modo che ogni parte (fold) funzioni da test set una volta. Nel mio caso specifico, il parametro k è uguale a 3 quindi questo processo incrociato è stato ripetuto per 3 volte.

Infine si confrontano le previsioni del modello con i dati reali e il programma genera una matrice di confusione che conta quanti elementi ha predetto nel giusto modo e quante volte si è confuso. È una matrice 3x3 (numero di azioni x numero di azioni) e l'ottimo è che la diagonale principale contenga valori prossimi al 100 la diagonale opposta contenga valori prossimi allo zero.

La matrice di confusione è utile per capire la precisione del nostro modello e per migliorarla si possono modificare alcuni parametri nel codice:

- Numero di k-folds (k): numero di volte per il quale si ripete il processo incrociato di training, validation e test
- Dimensione della finestra temporale (window_size): quanti secondi di dati vengono selezionati per fare una previsione
- Frequenza di campionamento (sampling_frequency): rappresenta la frequenza di registrazione dei dati da parte dei sensori
- Batch_size: numero di campioni elaborati dal modello in un passaggio di addestramento

- STEP 3: TESTING DEL MODELLO

Nell'ultima parte del progetto si procede con l'invio dello script main_test dove si valuta l'accuratezza del modello su nuovi dati. Il programma riceve i dati dal dispositivo relativi ai movimenti compiuti in tempo reale e, se il modello è preciso, riconosce l'azione svolta.

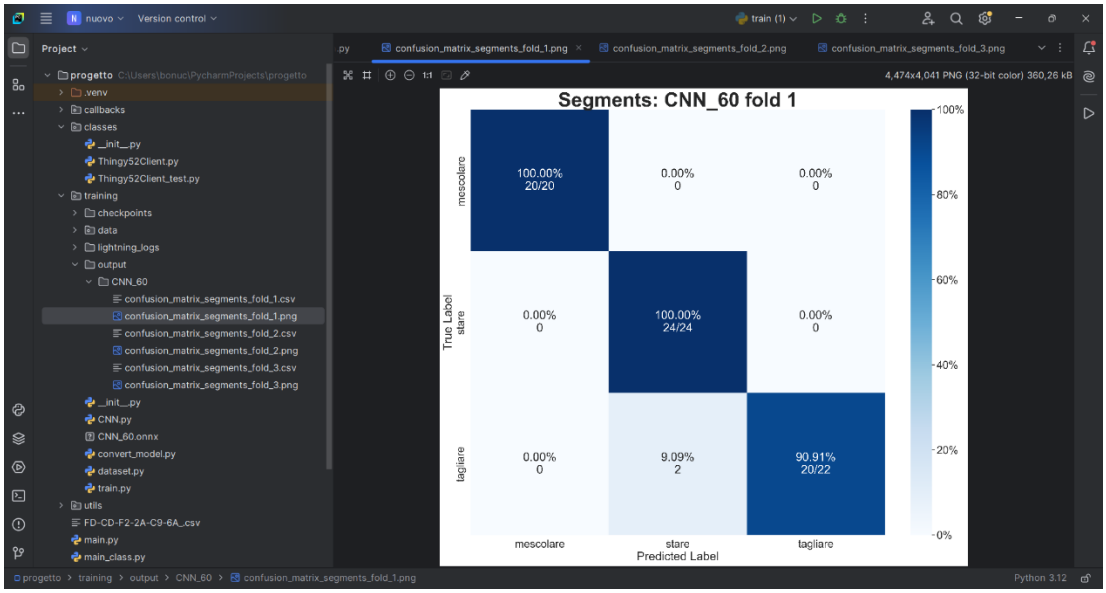
## • Il progetto

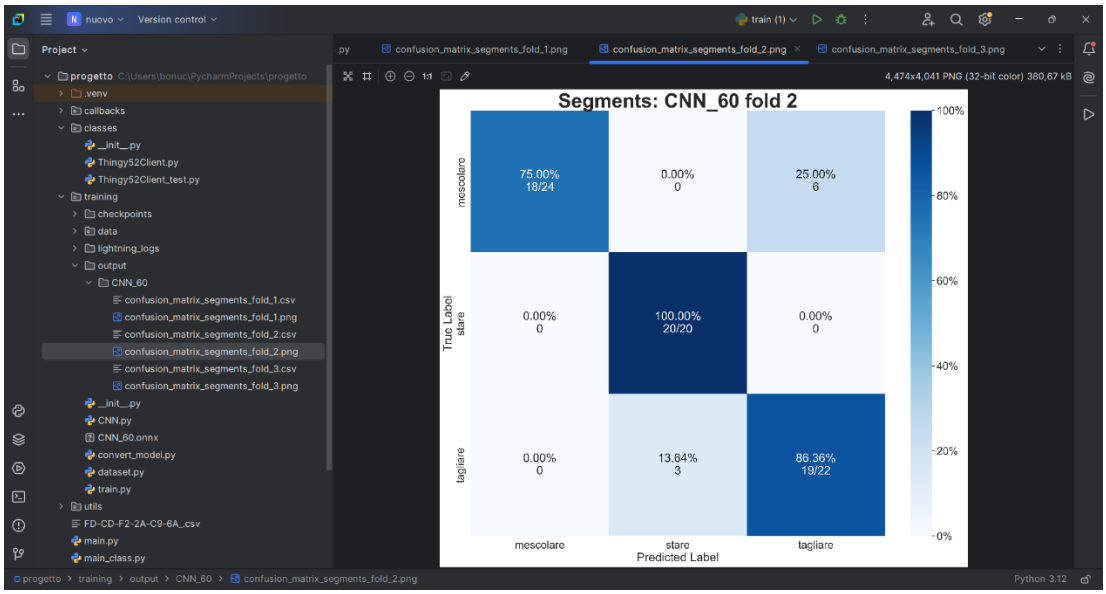
**PARAMETRI SCELTI**

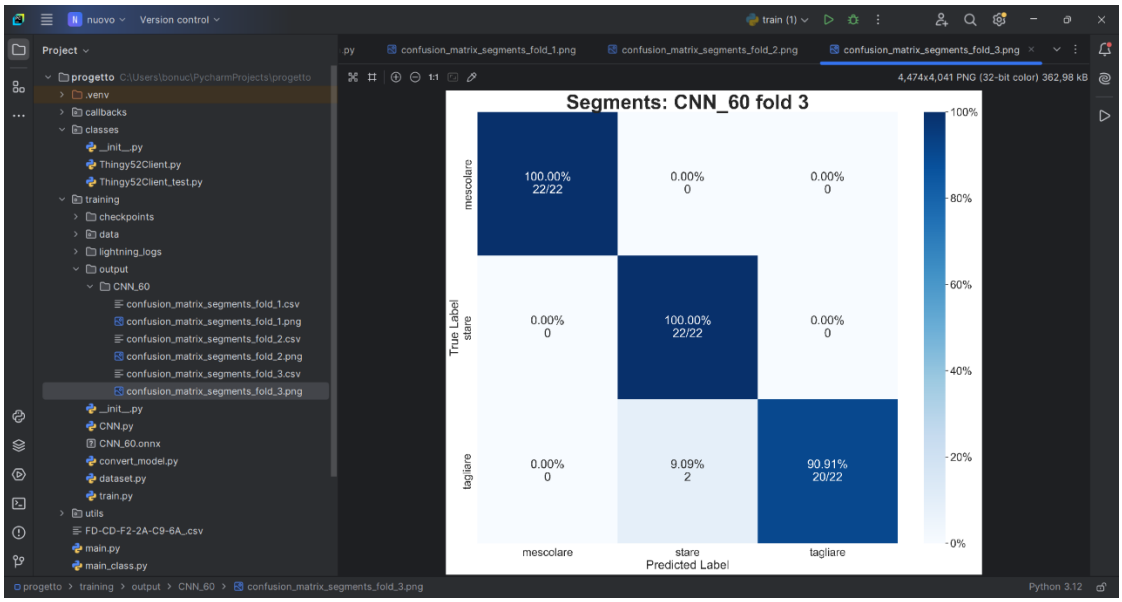
Come detto sopra, durante il training è possibile scegliere e modificare alcuni parametri per fare in modo di ottenere delle matrici di confusione il più possibile precise. Nel mio caso non è stato necessario modificare i parametri perché le matrici di confusione sono risultate sufficientemente precise al primo tentativo:

- k (k-folds): il parametro k è stato impostato su 3 quindi il processo incrociato è stato ripetuto 3 volte. Ho scelto il valore 3 in modo da non fare troppi cicli di addestramento ed evitare l'overfitting e quindi rendere il modello capace di generalizzare il più possibile.
- window_size: il valore di window_size è stato impostato a un secondo.
- sampling_frequency: la frequenza di campionamento è stata impostata 60 Hz.
- batch_size: il batch_size è stato impostato su 32 ossia il modello elabora 32 campioni di dati alla volta durante l'addestramento.

## MATRICI DI CONFUSIONE OTTENUTE



La matrice di confusione mostra buoni risultati e il modello riesce a identificare correttamente nella maggior parte dei casi le azioni di tagliare, mescolare e stare fermo.

Il risultato positivo potrebbe essere dovuto al fatto che:

1. Le azioni sono analizzate sono molto diverse tra di loro: tagliare implica un movimento lineare avanti e indietro, mescolare comporta un movimento circolare mentre lo “stare_fermo” non prevede movimento del Thingy52.
2. Differenza nell’utilizzo dei sensori: durante l'azione “tagliare” i dati dell’accelerometro in orizzontale sono i più importanti, durante l’azione “mescolare” i dati del giroscopio sono più determinanti mentre durante il gesto dello “stare_fermo” non abbiamo cambiamenti significativi di valore in nessun sensore.
3. Migliore riconoscimento dello “stare_fermo”: si nota che l’azione “stare_fermo” è quella meglio riconosciuta forse perché non prevede il movimento del Thingy mentre le altre due azioni hanno valori che variano in continuazione per quanto riguarda l’accelerometro e il giroscopio.

**TEST**

Nonostante le buone matrici di confusione, la fase di test con i dati in tempo reale presenta dei limiti. Infatti il modello fatica a riconoscere l’azione che si sta compiendo in tempo reale.

## • Conclusioni

Questo progetto ci ha dato l'opportunità di conoscere nuove aree della programmazione, tra cui l'interazione con il Thingy52 tramite BLE, la gestione dei dati dei sensori e l'applicazione di tecniche di machine learning per classificare le attività.