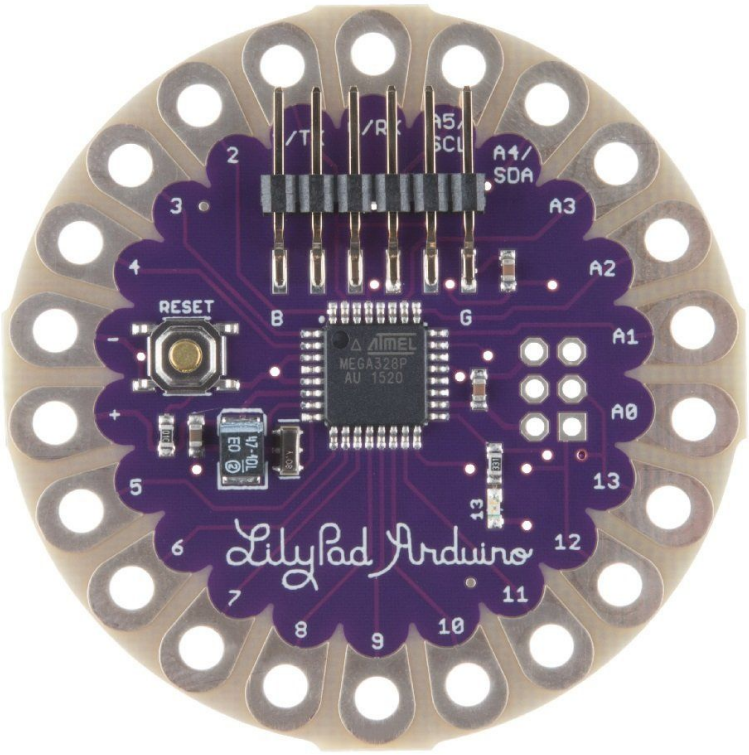


Ambient Assisted Living

Pietro Colombo 793679

Marco Fagioli 808176

Progetto Modelli Probabilistici per le Decisioni 2018-1019

Introduzione

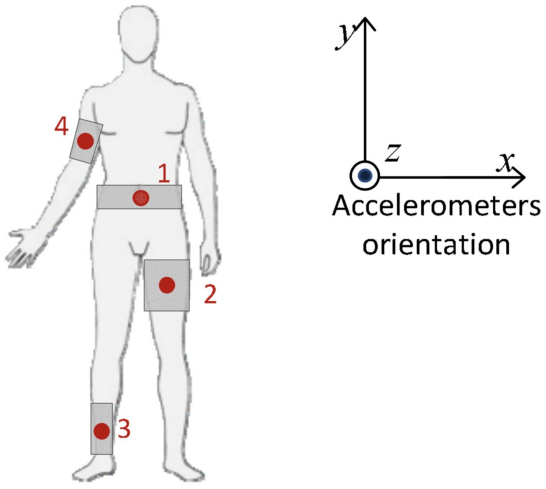
Recenti analisi demografiche mostrano che la vita media della popolazione si sta allungando. Questo porterà ad un incremento del bacino di utenza per le strutture sanitarie con ripercussioni sulla qualità dei servizi.

Ambient Assisted Living (AAL) è un progetto europeo per lo sviluppo di nuove tecnologie non invasive permettendo il monitoraggio delle attività quotidiane degli anziani in ambiente domestico, garantendo un prolungato stile di vita indipendente.

Questo lavoro ha come obiettivo la progettazione e lo sviluppo di un **modello predittivo** basato sulle misurazione fornite da quattro **sensori** installati sui soggetti. Gli accelerometri sono quattro e disposti:

1. sul **bacino**;
2. sulla **gamba** sinistra;
3. sulla **caviglia** destra;
4. sul **braccio** destro.

Scheme of positioning and orientation



Dataset

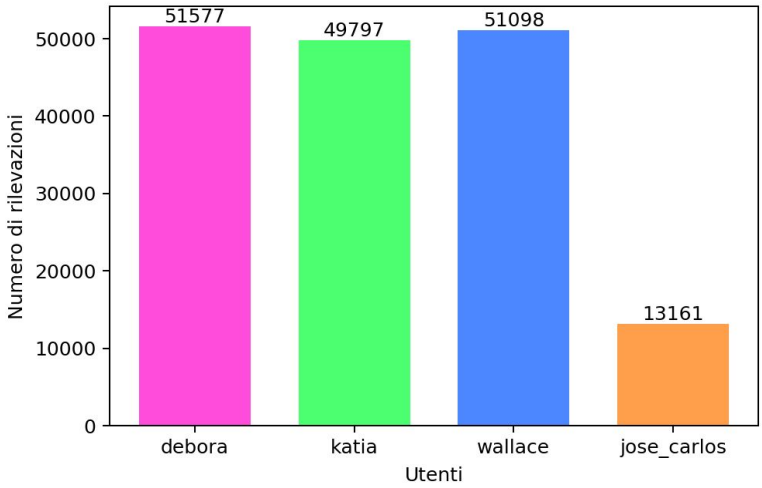
Il dataset è costituito da **165.633 rilevazioni** che sono state effettuate su **quattro persone** diverse per età e sesso, per un totale di due ore di misurazioni per ciascuna. Raggruppiamo le feature presenti all'interno del dataset in tre tipologie:

- dati anagrafici degli utenti;
    - **user**, nome dell'utente a cui è stata fatta la rilevazione;
    - **gender**, sesso dell'utente;
    - **age**, età dell'utente;
    - **how_tall_in_meters**, altezza dell'utente;
    - **weight**, peso dell'utente;
    - **body_mass_index**, indice di massa corporea dell'utente;
- misurazioni spaziali per **ogni sensore;**
    - **x**, accelerazioni sull'asse x;
    - **y**, accelerazioni sull'asse y;
    - **z**, accelerazioni sull'asse z;
- tipologia del movimento misurato;
    - **class,** etichetta che indica l'attività svolta dall'utente.

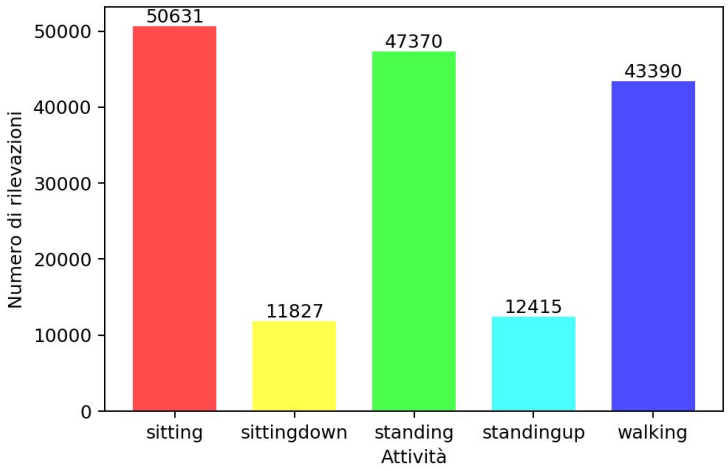
	user	gender	age	how_tall_in_meters	...	x4	y4	z4	class
0	debora	Woman	46	1,62	...	-150	-103	-147	sitting
1	debora	Woman	46	1,62	...	-149	-104	-145	sitting
2	debora	Woman	46	1,62	...	-151	-104	-144	sitting
3	debora	Woman	46	1,62	...	-153	-103	-142	sitting
4	debora	Woman	46	1,62	...	-153	-104	-143	sitting

Esplorazione Dataset

La prima operazione eseguita per analizzare il dataset è stata quella di andare a vedere come sono **distribuiti** gli utenti e le attività rispetto alle misurazioni.

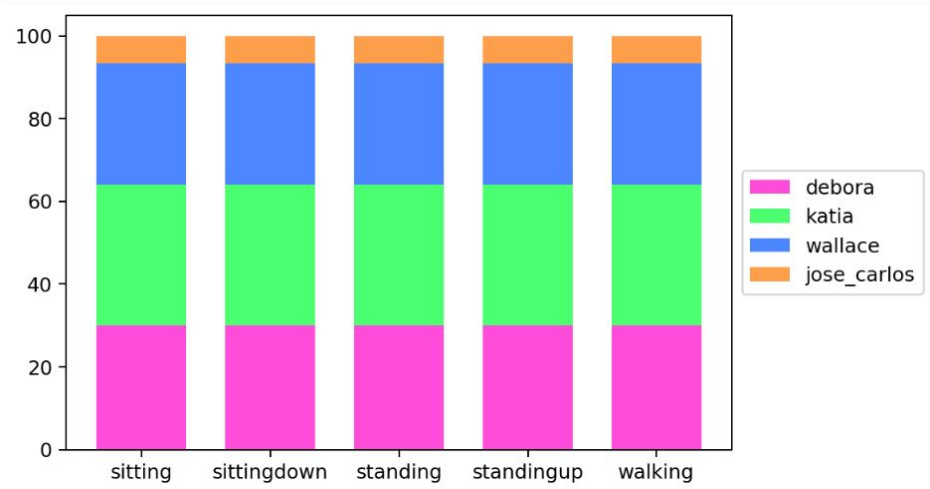


Il grafico evidenzia come il numero di misurazioni effettuato su *jose carlos* sia inferiore agli altri, probabilmente vista la maggior anzianità.



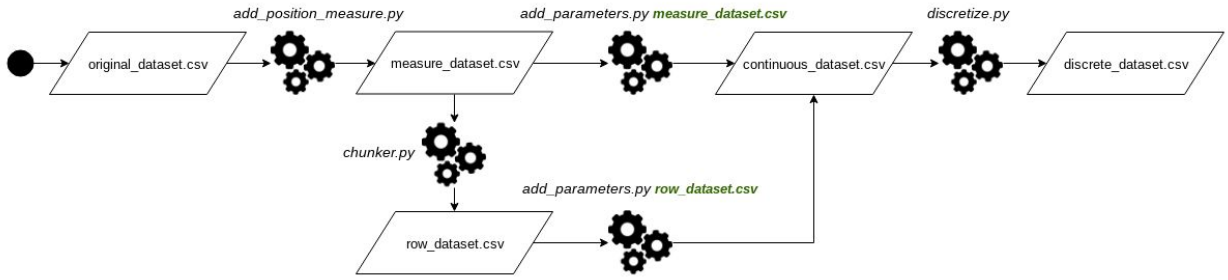
Il grafico evidenzia come il dataset contenga **tre attività** (sitting, standing e walking) con **alto numero**, circa 50.000, di misurazioni e le restanti due (sittingdown e standingup) con ⅛ delle **rilevazioni** effettuate.

L'ultima analisi è stata fatta riguardo la distribuzione degli utenti sulle misurazioni in base alle diverse attività. Si nota che per ogni tipologia di movimento svolta è stato **bilanciato** il numero di **rilevazioni** effettuate dagli utenti, fatta eccezione per jose carlos per i motivi citati sopra.



Data processing

Prima di poter progettare e sviluppare la modellazione del modello predittivo, sul **dataset originale** di partenza sono state eseguite delle **modifiche** per introdurre **nuovi parametri e discretizzare**. La sequenza di operazioni è descritta dal seguente **flowchart**:

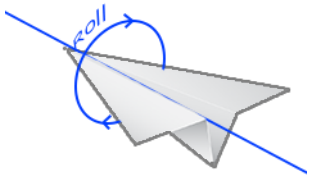


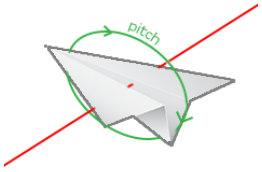
add_position_measure.py

Lo script si occupa di prendere il dataset iniziale e per ogni sensore calcola le misure di **roll** (rollio), **pitch** (beccheggio) e **accel** come norma delle accelerazioni.

Le formule utilizzate per i movimenti spaziali sono:

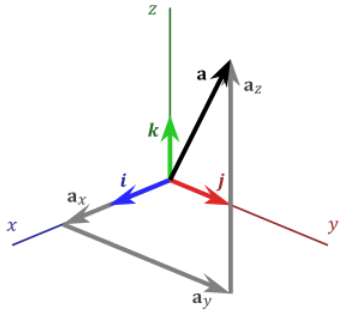
- $Roll = \frac{180}{\pi} * atan2(y, z)$





- $Pitch = \frac{180}{\pi} * atan2(-x, \sqrt{y^2 + z^2})$

- $Accel = \sqrt{x^2 + y^2 + z^2}$



chuncker.py

Con questo script, visto che 8 tuple del dataset corrispondono circa alle **informazioni** delle rilevazioni in un **secondo**, è stata calcolata su di esse la **varianza** dei roll e dei pitch, e la **norma** delle accelerazione per ognuno dei quattro

sensori. L'operazione è stata effettuata per avere una miglior misura riguardo il moto e movimento che l'utente stesse effettuando. Al fine del funzionamento del modello è anche possibile evitare di eseguire questo script e di lavorare sul dataset con la totalità delle sue righe. (N.B. L'approccio senza passare per le varianze e le norme risulta però peggiore e si ottiene un modello con prestazioni inferiori).

add_parameters.py

Lo script prevede in input il dataset sul quale essere utilizzato per poter essere adattato in base al percorso di elaborazione seguito.

L'elaborazione si occupa di calcolare la media e la deviazione standard per la norma delle accelerazioni tra i vari sensori, inoltre per ogni classe viene creata una feature che identifica la classe con 1 quando è presente e con 0 altrimenti.

discretize.py

Lo script si occupa di effettuare la discretizzazione su tutte le feature di interesse; essa viene fatta su 10 bins, parametro che è stato scelto dopo varie sperimentazioni ed è quello che ha garantito una migliore rappresentazione della funzione di densità.

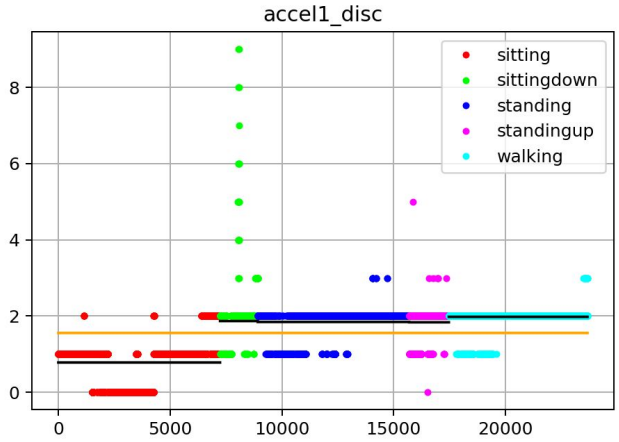
Il dataset trattato, modificato ed elaborato, al termine del processo è nella forma:

	accel1	accel2	accel3	...	accel4_disc	accel_mean_disc	accel_std_disc
0	113.982044	29.443006	137.710577	...	0	0	1
1	114.797920	28.398393	137.909730	...	0	0	1
2	114.892001	28.135276	137.932841	...	0	0	1
3	115.612094	28.280680	138.280399	...	0	0	1
4	118.338101	28.052128	137.944565	...	1	0	1

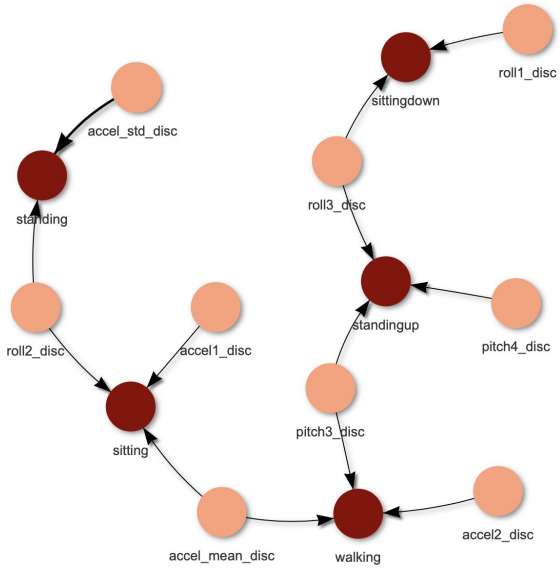
Modellazione rete bayesiana

Primo approccio

Come prima idea per la modellazione della rete bayesiana si è deciso di effettuare un'analisi della **correlazione** tra le feature di interesse e le classi dei movimenti. L'analisi è stata effettuata sui campi di rilievo per cui è stato calcolato il valore medio associato ad ogni classe di **movimento**. Alla ricerca della correlazione tra campo e movimento è stato creato un grafico come il seguente esempio.

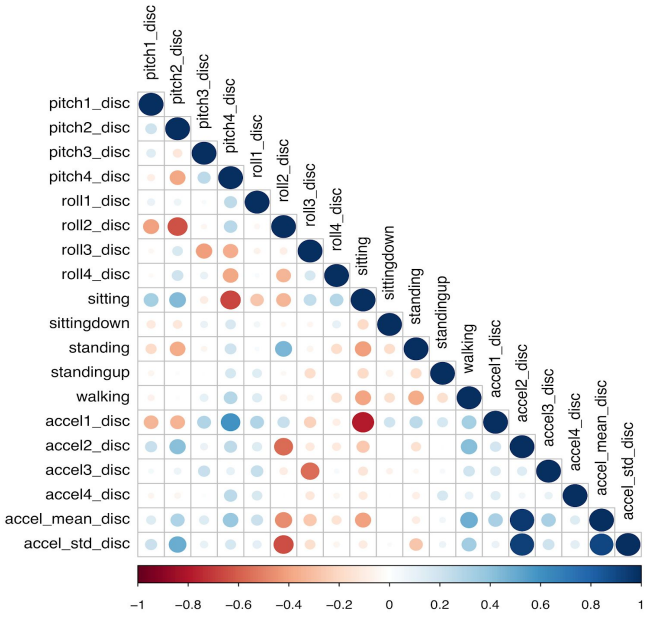


L'analisi con questo metodo su ogni feature ha portato a generare la rete:

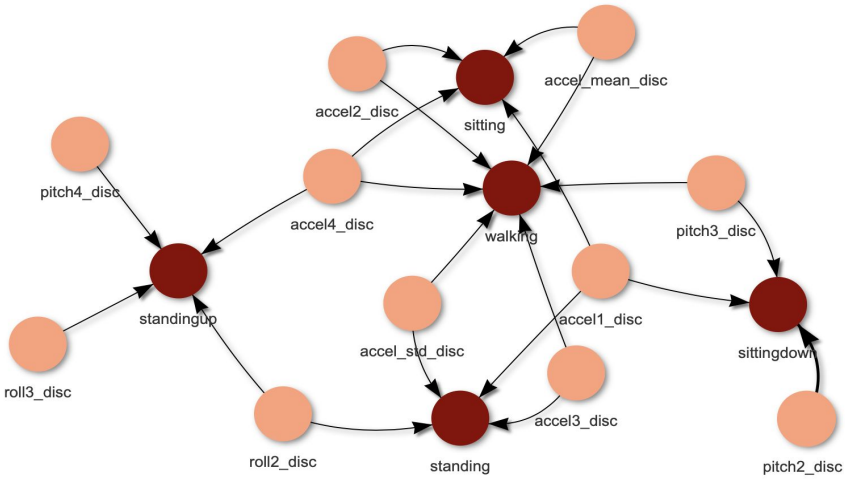


Correlazione di Pearson

Come secondo approccio per la generazione della rete bayesiana è stata usata la funzione *cor* della libreria *caret*, che dato in input un dataset computa e restituisce una matrice in cui viene calcolato il **coefficiente** di **correlazione** di **Pearson** per ogni coppia di feature che compone il dataset.



Ognuna delle feature è stata analizzata e nei casi in cui con una determinata classe di movimenti ci fosse una correlazione particolare usata, generando la rete:



Implementazione rete bayesiana

Come operazione preliminare è necessario effettuare la divisione del dataset in parte di **training** e di **testing**. Questo passaggio è computato dallo script *split_dataset.py* che divide il dataset in 80% train e 20% test, mantenendo **bilanciato** il numero di elementi per ogni tipologia di movimento.

La parte di implementazione del **modello predittivo** per la rete bayesiana è stata effettuata tramite l'ambiente di sviluppo *R*, in particolare usando il package *bnlearn.*

Inizialmente è necessaria una definizione manuale della **struttura** della **rete** e perciò vengono creati ed istanziati prima i nodi e poi collegati con gli archi necessari. Sono poi caricati i dataset di train e di test, e viene effettuato l'apprendimento e fitting del modello, tramite il metodo *bayes*, sui dati di training.

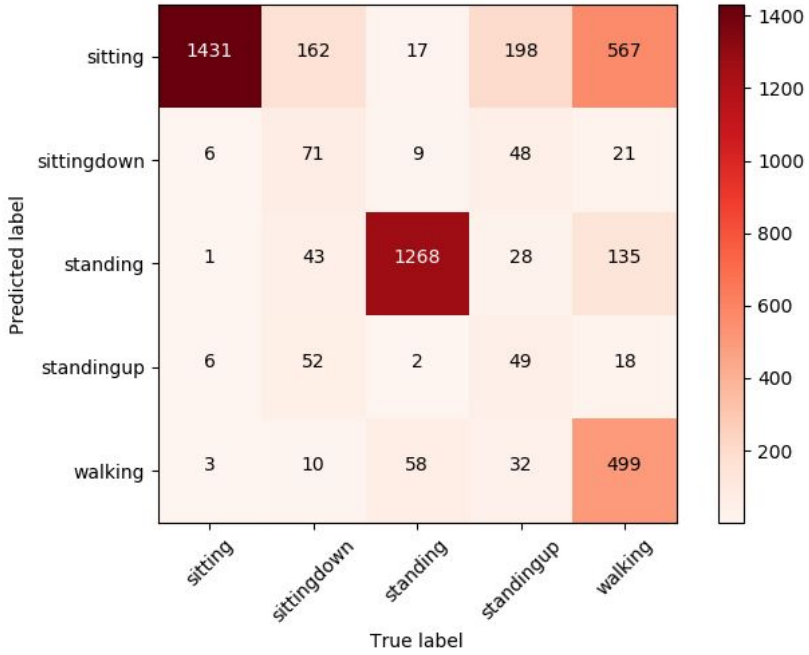
La parte di previsione del dataset di testing è effettuata mediante **inferenza**. Iterativamente viene presa una riga alla volta:

- vengono estratte le feature di interesse;
- per ognuno dei nodi associato alle classi dei movimenti viene effettuata l'inferenza esatta;
- si sceglie il movimento con probabilità calcolata maggiore e viene confrontato con il target, andando a costruire man mano la matrice di confusione.

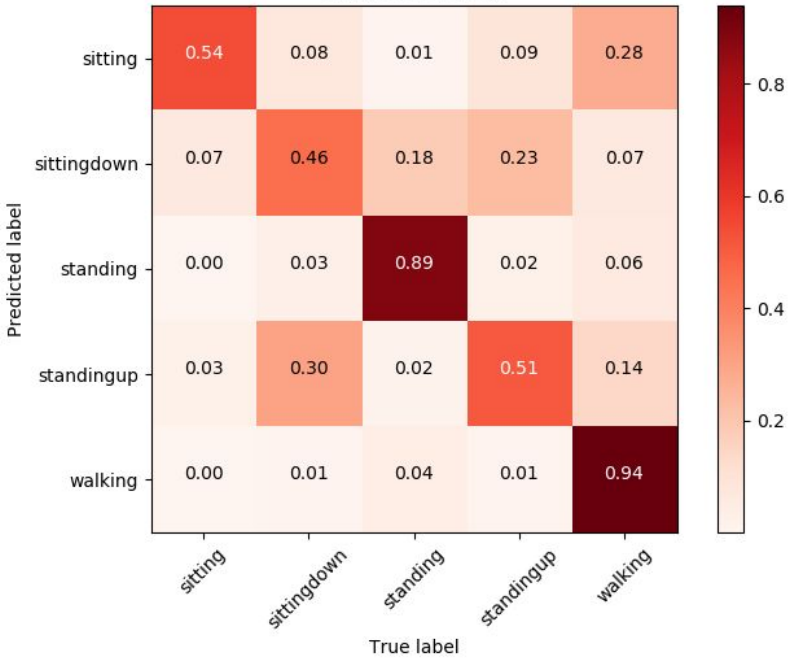
È possibile visualizzare la matrice di confusione tramite una stampa con un codice python, vengono inoltre valutate le metriche associate.

Risultati

La matrice di confusione ottenuta con il primo modello generato è la seguente:



La matrice di confusione ottenuta con il secondo modello generato è la seguente:



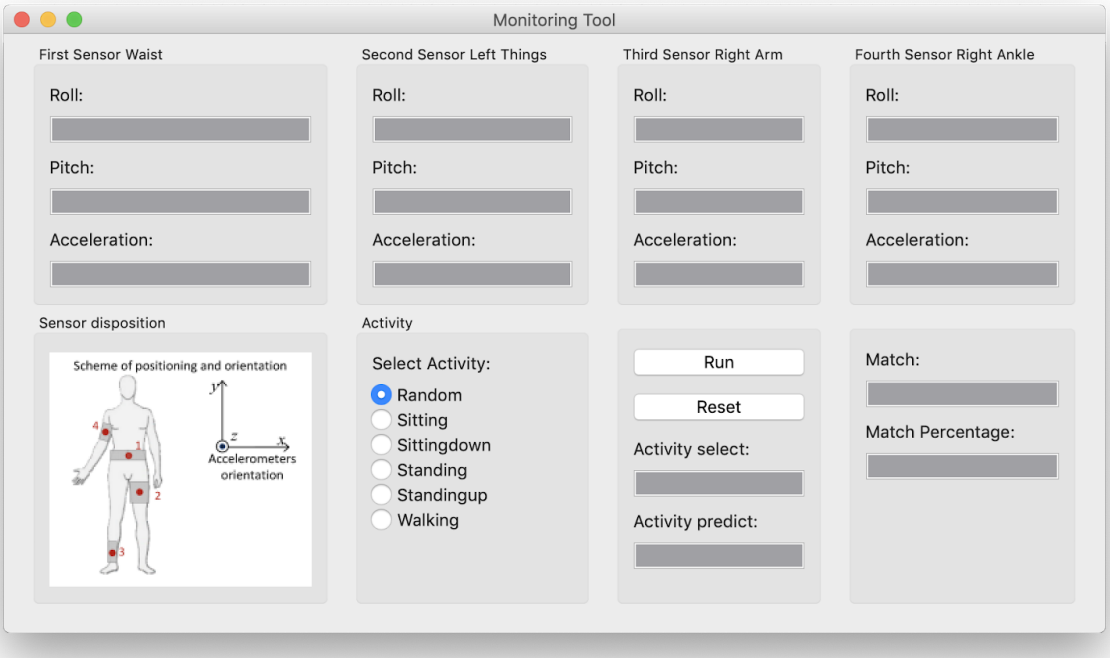
Entrambi i modelli creati hanno delle performance molto simili, si riscontra una lieve differenza nelle accuracy, il primo ha un **accuracy** del **70%** mentre il secondo del **68%**.

Confrontiamo le misure di **recall** e **precison** per i due modelli:

	sitting	**sittingdown**	**standing**	**standingup**	**walking**
	Primo modello				
recall	0.989	0.210	0.936	0.138	0.402
precision	0.603	0.458	0.860	0.386	0.829
	Secondo modello				
recall	0.992	0.142	0.953	0.172	0.316
precision	0.543	0.457	0.888	0.513	0.940

Le due reti generate hanno **performance** molto **simili**, è stato deciso per l'interfaccia di utilizzare la prima vista la maggior semplicità della rete e leggera accuratezza in più.

Graphical User Interface



L'interfaccia è realizzata in Python utilizzando **PyQt5**, permette di selezionare **l'attività** per la quale si vuole fare inferenza sulla rete addestrata.

Premendo ***Run*** viene estratto casualmente dal dataset di test una tupla della attività selezionata e viene fatta inferenza chiamando lo script R "*script_for_gui.r*".

Nella parte in basso a destra vengono visualizzate le **statistiche** delle inferenze effettuate.