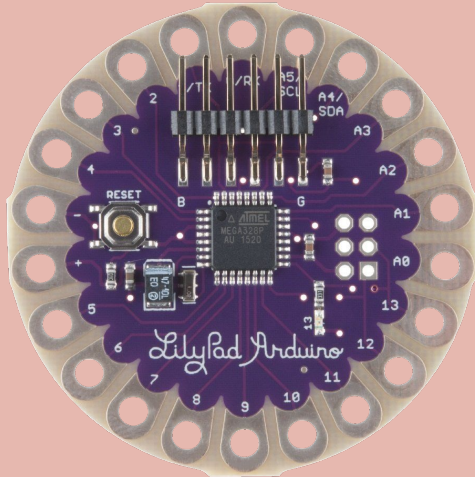


# Ambient Assisted Living



Pietro Colombo **793679**  
Marco Fagioli **808176**

# Ambient Assisted Living

Ambient Assisted Living è un programma di **ricerca** europeo verso lo sviluppo di **tecnologie** per **l'assistenza** agli anziani in ambiente domestico con la finalità di:

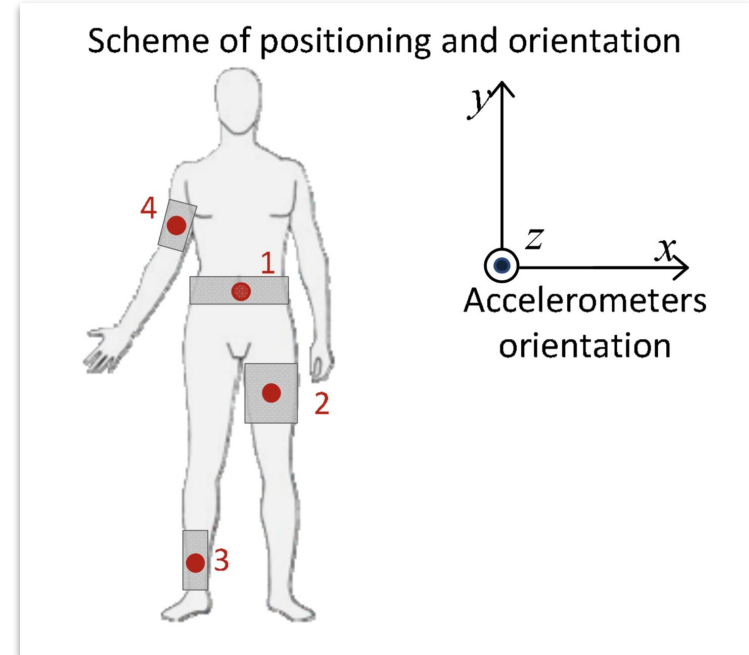
- migliorare la loro **autonomia**;
- facilitare le attività **quotidiane**;
- garantire buone condizioni di **sicurezza**;
- **monitorare** e **curare** le persone malate.



# Obiettivi del progetto

Sviluppo e progettazione di un **modello predittivo** sulle misurazioni di 4 **sensori**:

- applicati su bacino, gamba, caviglia e braccio;
- effettuate su 4 **persone** con età e sesso differenti;
- registrate circa 8 **rilevazioni** per secondo.



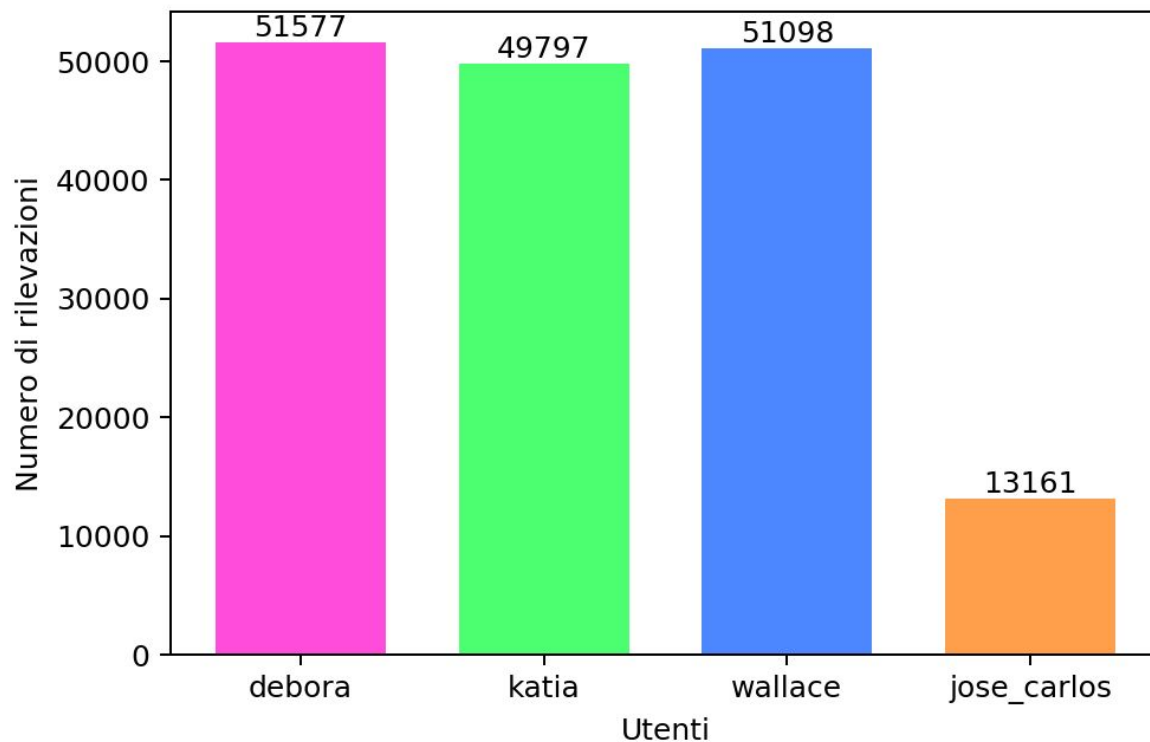
# Dataset

Il dataset contiene **165.633 rilevazione** definite su 19 **feature**:

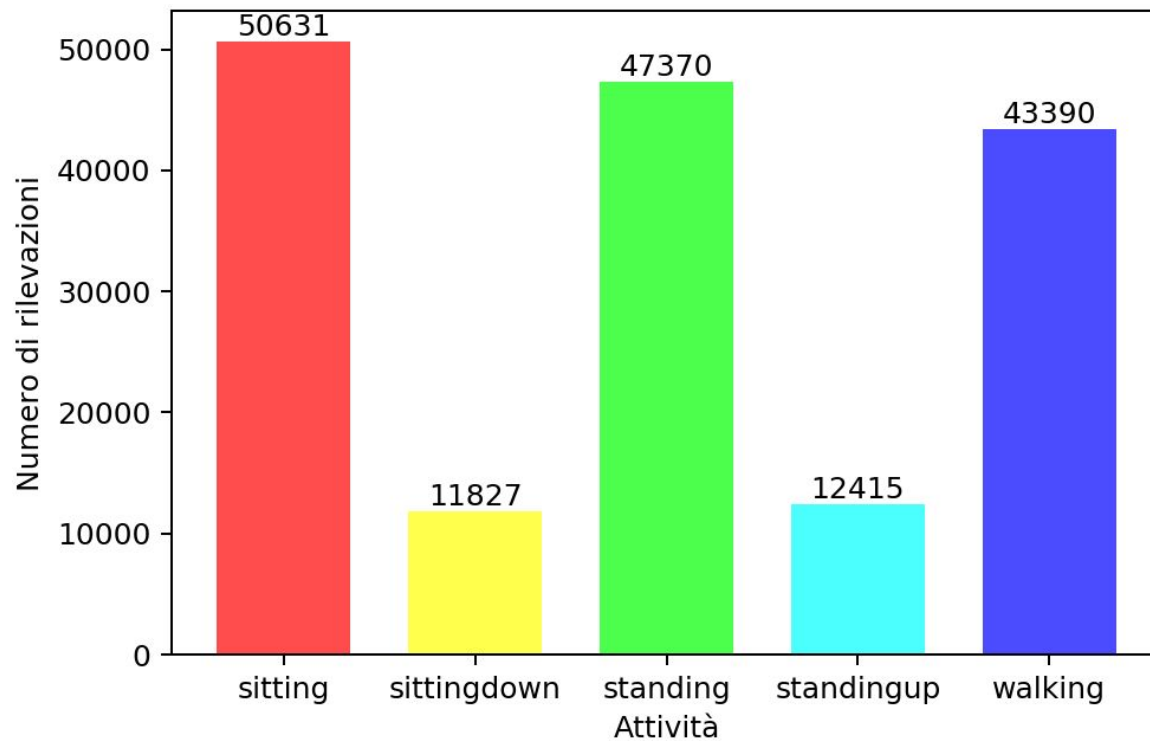
- dati anagrafici degli utenti
- tipo di movimento effettuato
- per ogni sensore una misurazione nello spazio x, y e z

	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

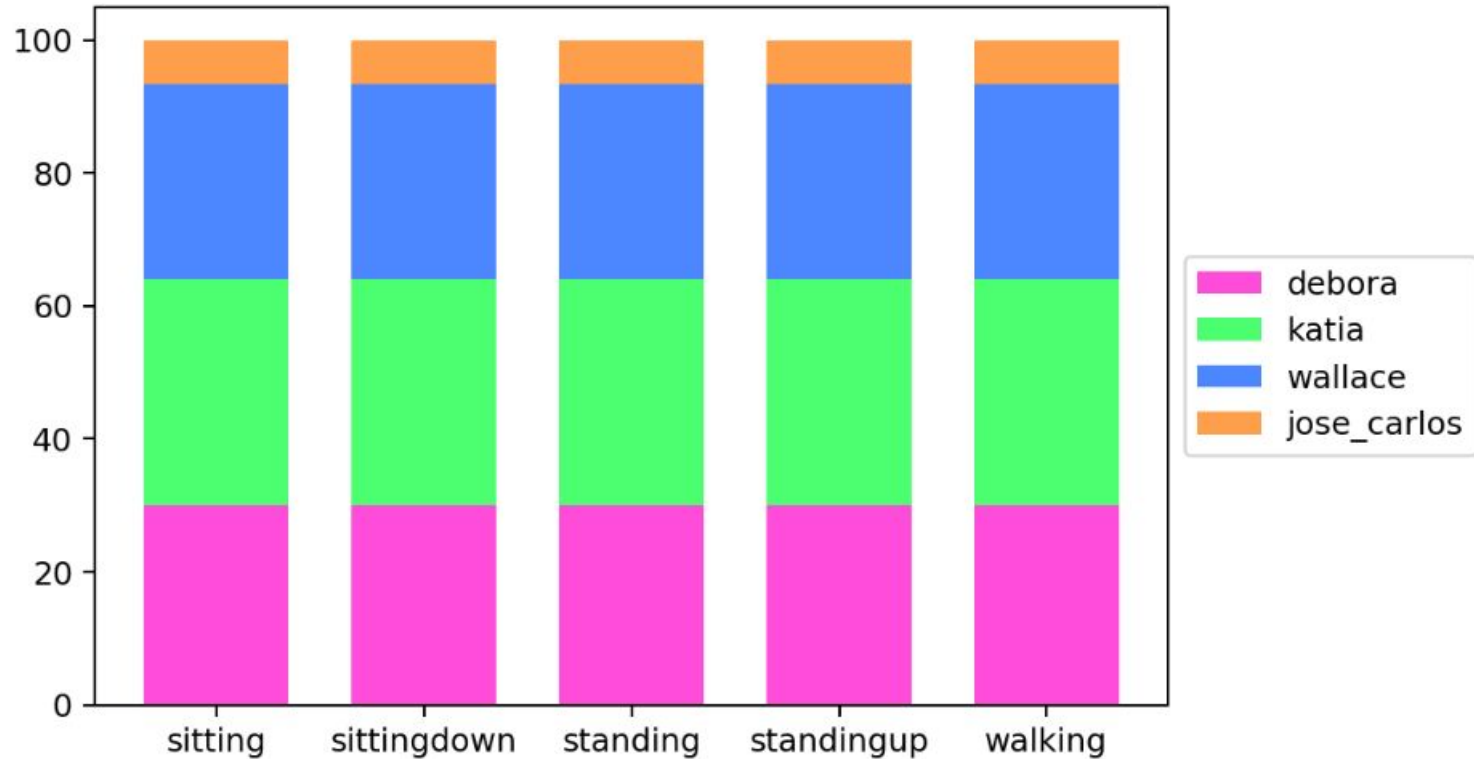
# Distribuzione rilevazioni per utente



# Distribuzione rilevazioni per classe

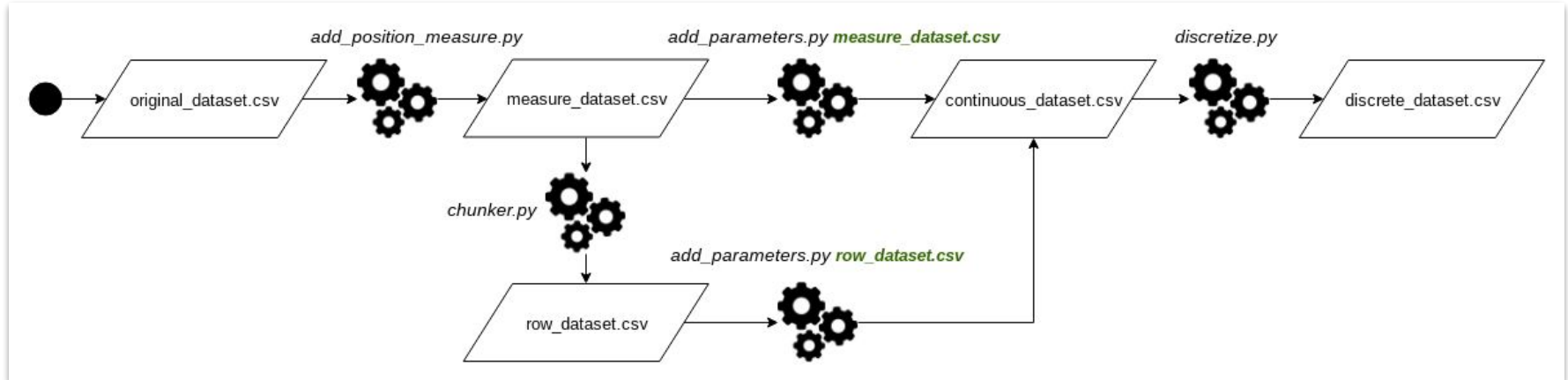


# Distribuzione attività per utente



# Data preprocessing

La progettazione e sviluppo del **modello predittivo** ha richiesto un **elaborazione del dataset**, la sequenza di operazioni è descritta dal seguente flowchart:



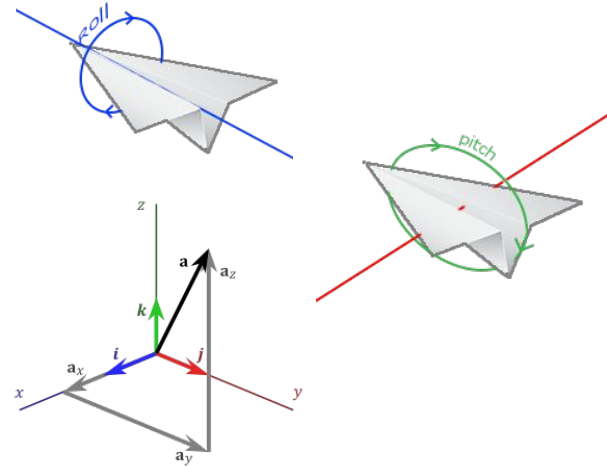


# Data preprocessing

`add_position_measure.py`

Per ogni sensore e riga del dataset vengono calcolati:

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



# Data preprocessing



Le successive operazioni step by step sono:

- **chuncker.py** (opzionale), unisce il dataseta chunk di 8 righe su cui calcola variazioni e moduli dei parametri;
- **add\_parameters.py**, calcola media e deviazione standard delle accelerazioni dei sensori e costruisce una feature per ogni movimento;
- **discretize.py**, discretizza tutte le feature di interesse su 10 intervalli.

# Dataset finale

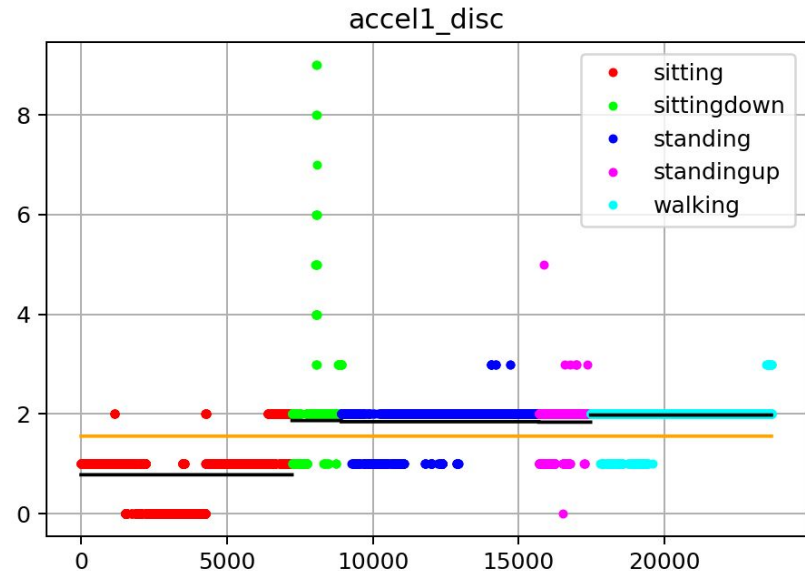


	accel1	accel2	accel3	...	accel4_disc	accel_mean_disc	accel_std_disc
0	113.982044	29.443006	137.710577	...	0	0	1
1	114.797920	28.398393	137.909730	...	0	0	1
2	114.892001	28.135276	137.932841	...	0	0	1
3	115.612094	28.280680	138.280399	...	0	0	1
4	118.338101	28.052128	137.944565	...	1	0	1

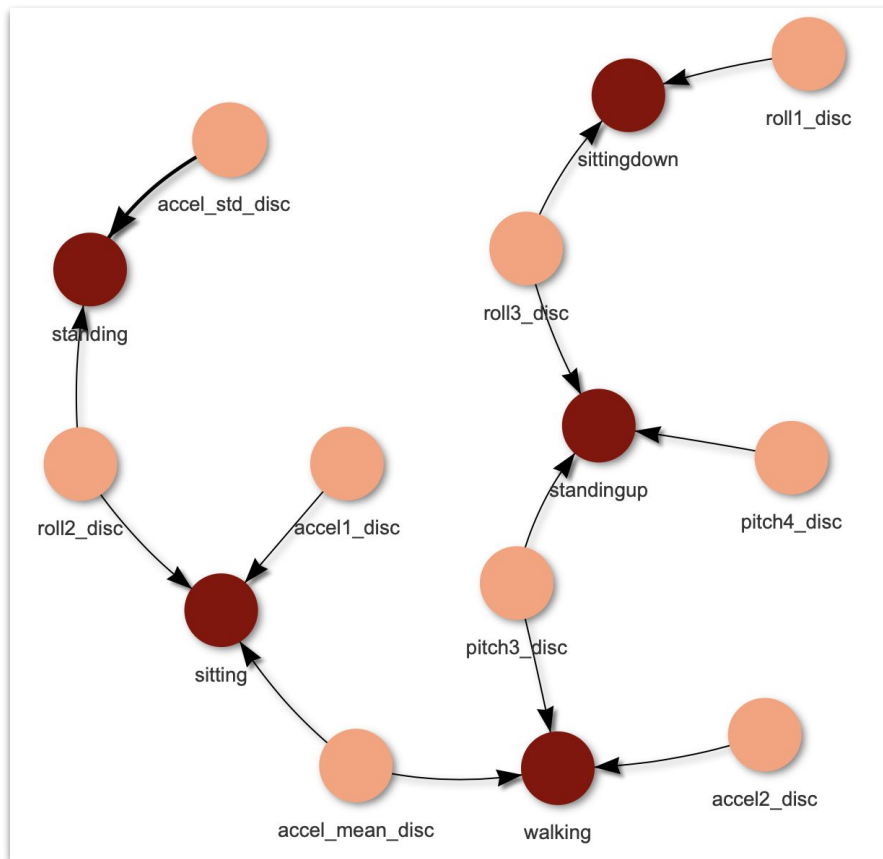
# Prima modellazione rete bayesiana

Iniziale analisi della **correlazione** tra le **feature** di interesse e le classi dei **movimenti**.

È stata effettuata sui campi di rilievo per cui è stato calcolato il **valore medio** associato ad ogni classe di movimento.



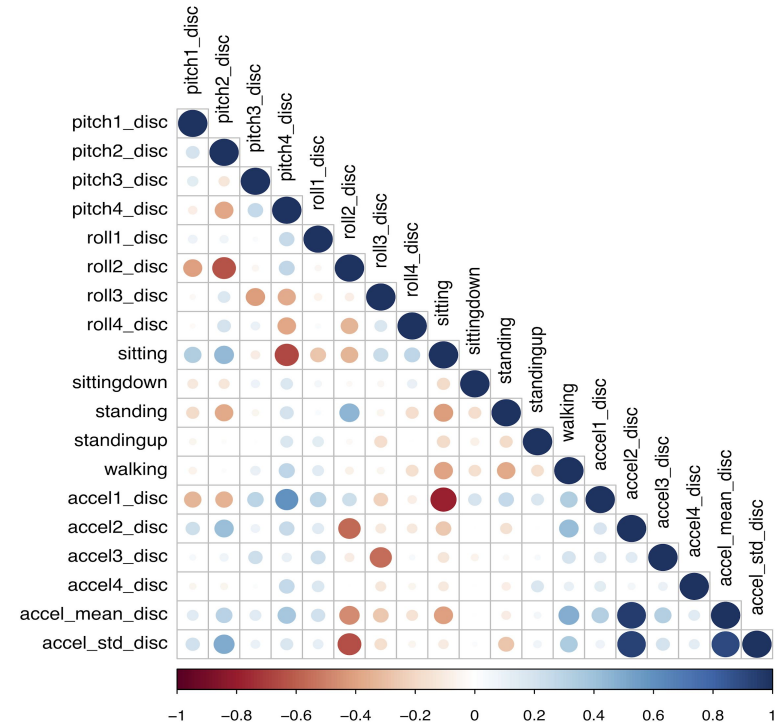
# Prima Rete Bayesiana



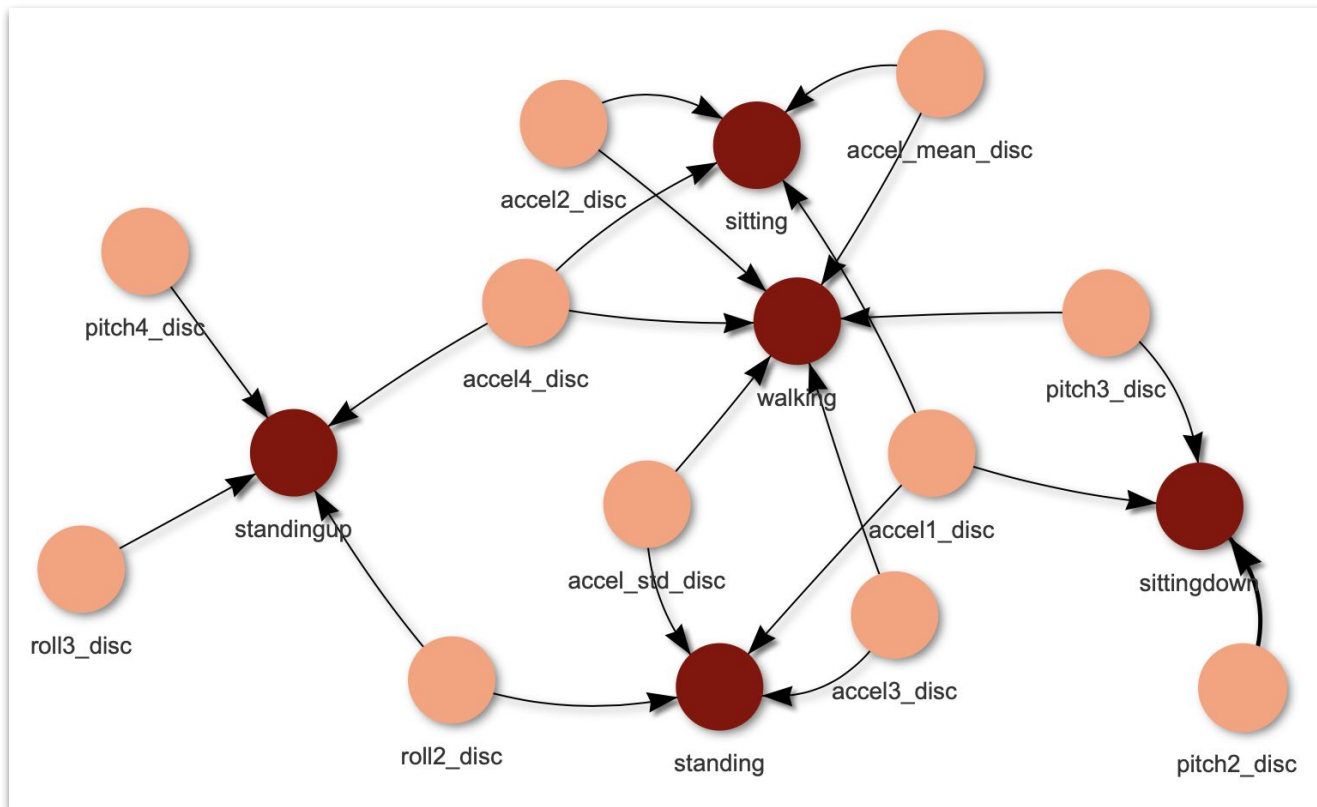
# Modellazione rete con correlazione di Pearson

Funzione `cor` della libreria `caret`

- **input:** dataset definito su più colonne
- **output:** matrice in cui viene calcolato il coefficiente di correlazione di Pearson per ogni coppia di feature



# Rete Bayesiana con correlazione di Pearson



# Implementazione rete bayesiana

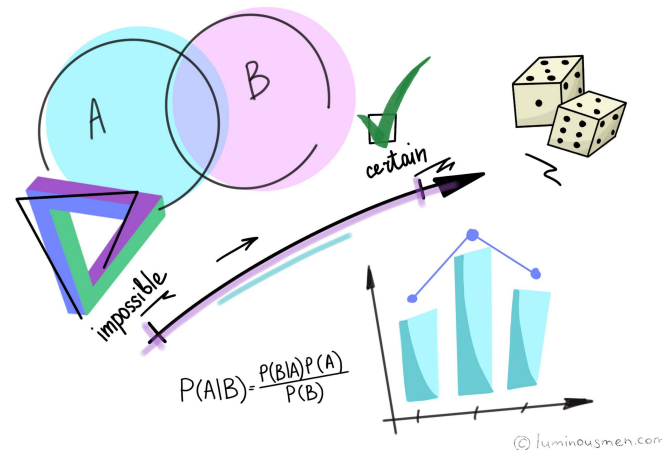
- Divisione del dataset in **80% train** e **20% test**, mantenendo **bilanciato** il numero di elementi per ogni tipologia di movimento.
- Implementazione del **modello predittivo**
  - effettuata tramite l'ambiente di sviluppo R;
  - usando il package `bnlearn`.





# Creazione e fitting del modello

- Iniziale **definizione** manuale della struttura della **rete**
  - creazione dei nodi ed archi
- Vengono caricati i dataset di train e di test
- Effettuato l'**apprendimento** e **fitting** del modello
  - usato il metodo bayes



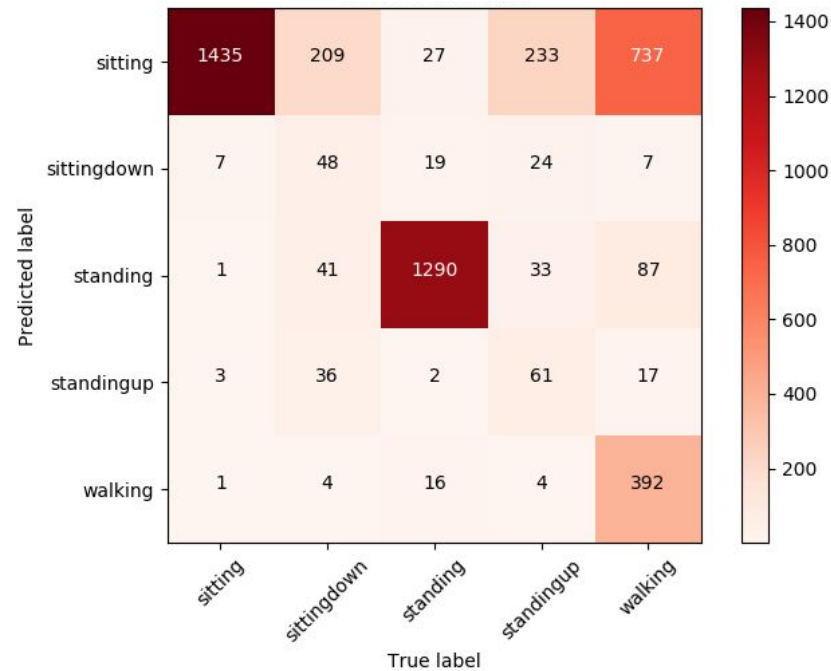
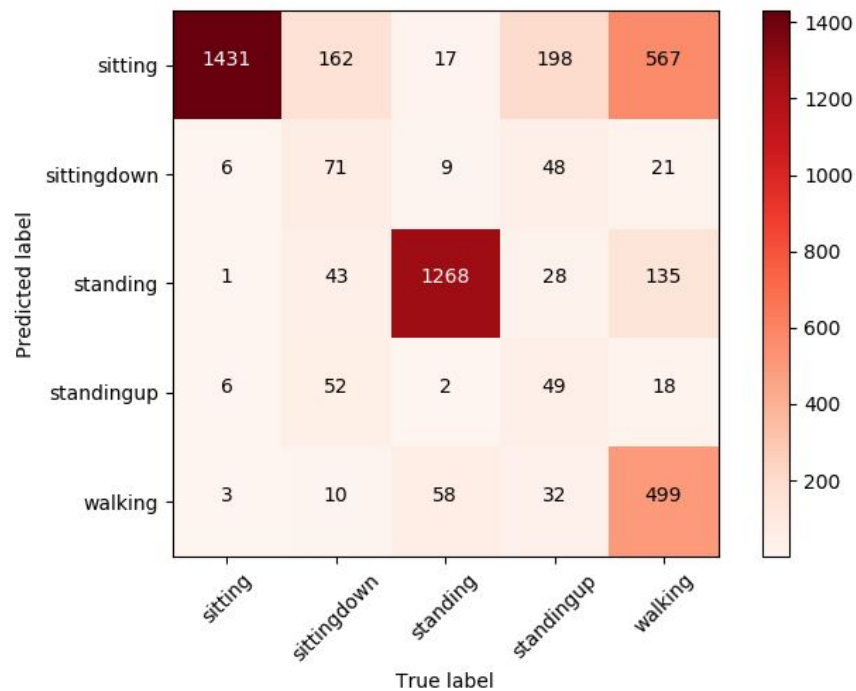
# Inferenza



Per la previsione sul dataset di test viene usata l'inferenza. Per ogni riga:

- 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**.

# Prima rete vs rete Pearson



# Risultati

	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 reti generate hanno **performance simili**. Per l'interfaccia viene usata la prima vista la maggior semplicità e leggera accuratezza in più, **70%** invece che 68%.

# Graphical User Interface

Monitoring Tool

First Sensor Waist

Roll:

Pitch:

Acceleration:

Second Sensor Left Thighs

Roll:

Pitch:

Acceleration:

Third Sensor Right Arm

Roll:

Pitch:

Acceleration:

Fourth Sensor Right Ankle

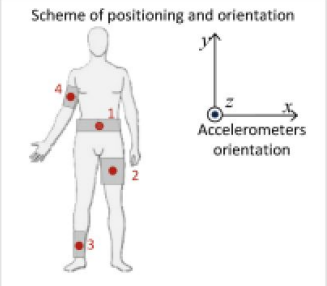
Roll:

Pitch:

Acceleration:

Sensor disposition

Scheme of positioning and orientation



Activity

Select Activity:

- ☒ Random
- ☐ Sitting
- ☐ Sittingdown
- ☐ Standing
- ☐ Standingup
- ☐ Walking

Run

Reset

Activity select:

Activity predict:

Match:

Match Percentage:

# Graphical User Interface

Tramite l'interfaccia è possibile selezionare **un'attività** su cui fare l'inferenza.

L'apposito pulsante **chiama** `script_for_gui.r`, per l'inferenza su un nuovo input.

Activity

Select Activity:

- ☒ Random
- ☐ Sitting
- ☐ Sittingdown
- ☐ Standing
- ☐ Standingup
- ☐ Walking

Run

Reset

Activity select:

Activity predict:

Match:

Match Percentage:

Vengono visualizzate le **statistiche** delle **inferenze** effettuate.

# Grazie per l'attenzione!

Pietro Colombo  
Marco Fagioli

