Università degli Studi Milano-Bicocca

Corso di Laurea Magistrale in Informatica

Progetto di Modelli Probabilistici per le Decisioni

- Relazione Finale -

Amrani Hamza *mat. 807386*
Brumana Mattia *mat. 808374*
Carta Costantino *mat. 808417*

Indice

Capitolo 1

Introduzione

Con l'aumento della speranza di vita e l'invecchiamento della popolazione, sviluppare nuove tecnologie che consentano ad anziani e malati una vita più indipendente e sicura è diventata una sfida. *AAL* (*Ambient Assisted Living*) è una possibilità per aumentare l'indipendenza e ridurre i costi di trattamento, ma sono necessarie ulteriori conoscenze al fine di sviluppare applicazioni informatiche onnipresenti che forniscano supporto all'assistenza domiciliare e consentano una maggiore collaborazione tra medici, famiglie e pazienti. *HAR* (*Human Activity Recognition*) è un'area di ricerca attiva il cui obiettivo è favorire lo sviluppo di tecnologie assistive per sostenere anziani, malati e persone con bisogni speciali. L'attività di riconoscimento può essere utilizzata per fornire informazioni sulle abitudini dei pazienti e, successivamente, per lo sviluppo di sistemi e-health (come AAL). A questo scopo, HAR utilizza due approcci: elaborazione di immagini e utilizzo di sensori indossabili da soggetti umani.
Negli ultimi anni, la ricerca sul riconoscimento di attività umane ha ottenuto buoni risultati; di conseguenza, HAR viene considerata come potenziale tecnologia per sistemi e-health.
Si propone quindi un classificatore HAR basato sull'apprendimento automatico, creato a partire da una completa descrizione sperimentale contenente le informazioni sui dispositivi HAR (*accelerometri*) e da un dataset di dominio pubblico contenente 165.633 campioni. In particolare, si considerano 5 classi di attività raccolte da 4 soggetti che indossano rispettivamente 4 accelerometri posizionati sul bacino (1), sulla coscia sinistra (2), sul braccio destro (3) e sulla caviglia destra (4) (Figura 1.1).

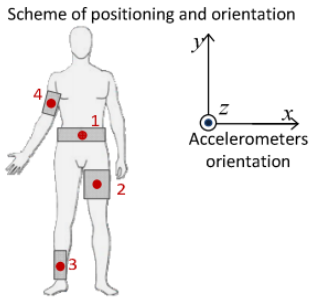

Figura 1.1: Posizione degli accelerometri utilizzati.

Tutti gli accelerometri sono stati calibrati prima della raccolta dati. Tale calibrazione consiste nel posizionare i sensori e nel rilevarne i valori da considerare come "zero". Dalla calibrazione, i valori letti di ciascun asse durante la fase di raccolta dati vengono sottratti dai valori ottenuti al momento della calibrazione. Al fine di attenuare i problemi peculiari di inaccuratezza di questo tipo di sensore, la calibrazione iniziale è stata eseguita nella stessa posizione sulla parte superiore di un tavolo piatto. Successivamente per soggetto, leggendo i dati degli accelerometri e calibrandoli dopo essere stati indossati dallo stesso. Ciò può avvantaggiare la raccolta dei dati, consentendo di ottenerne una maggiore omogeneità. Tuttavia, rende più complesso il riutilizzo del dispositivo dopo che è stato indossato da un particolare soggetto.

Lo scopo del progetto è creare un modello di apprendimento di parametri in grado di riconoscere autonomamente il movimento effettuato dal soggetto (*sitting, standing, sitting down, standing up, walking*) date le coordinate situate nello spazio dei quattro accelerometri.

Capitolo 2

Scelte implementative

Il progetto è stato implementato in Python, linguaggio di programmazione open-source ad alto livello, ottimo per il calcolo scientifico.
In particolare sono state utilizzate le seguenti librerie:

- *pandas*: per l'analisi dei dati e la creazione del dataset;
- *pgmpy*: per la generazione della rete bayesiana e l'inferenza;
- *pomegranate*: pacchetto alternativo per la generazione della rete bayesiana;
- *matplotlib*: per la creazione di grafici.

2.1 Pandas

Pandas è una libreria software scritta per il linguaggio di programmazione Python per la manipolazione e l'analisi dei dati. In particolare, offre strutture dati e operazioni per manipolare tabelle numeriche e serie temporali.
La libreria *pandas* risulta particolarmente utile al nostro scopo in quanto fornisce tipi di dati 2D (chiamati *DataFrame*). Un DataFrame è una tabella di oggetti eterogenei contenente indici sia per le righe (*index* rappresenta le etichette delle righe) che per le colonne (*columns* rappresenta le etichette delle colonne).
I dati presenti nel dataset vengono copiati in una DataFrame, in modo da agevolarne l'elaborazione e la manipolazione.

2.2 pgmpy

Pgmpy è una libreria Python utilizzata per l'implementazione di modelli grafici probabilistici e relativi algoritmi di inferenza e apprendimento. E' stato possibile rea-

lizzare una rete Bayesiana grazie alle seguenti funzioni messe a disposizione da tale libreria:

- *model = BayesianModel(edges)*: classe base per il modello Bayesiano. Il modello memorizza nodi e archi con distribuzione di probabilità condizionanta (*cpd*) e altri attributi. Gli archi vengono passati in input alla funzione come dati per inizializzare il grafo;
- *model.fit(train)*: stima le *cpd* di ogni variabile secondo un dato dataset (nel nostro caso, il dataset preso in considerazione è quello di *train*).

2.3 Pomegranate

Pomegranate è un pacchetto Python che implementa modelli probabilistici veloci e flessibili, che vanno dalle distribuzioni di probabilità individuali a modelli compositivi come *Bayesian Networks* e *Hidden Markov Models*. Alla base di Pomegranate vi è l'idea che tutti i modelli possano essere visti come una distribuzione di probabilità, in quanto tutti forniscono stime di probabilità per i campioni e possono essere aggiornati dati i campioni e i loro pesi associati. Da ciò ne consegue che i componenti implementati in Pomegranate possono essere impilati in modo più flessibile rispetto ad altri pacchetti.
Tramite Pomegranate è quindi possibile creare una Rete Bayesiana o un Hidden Markov Model che fanno previsioni sulle sequenze.
Abbiamo provveduto a creare una Rete Bayesiana partendo dal dataset finale (illustrato nel Capitolo 4) utilizzando semplicemente le seguenti istruzioni:

- *model = BayesianNetwork.from_samples(train, algorithm='greedy', state_names=header)*: apprende la struttura della rete dai dati (letti dal dataset *train*);
- *model.bake()*: finalizza la topologia del modello;
- *model.plot()*: disegna il grafo del modello.

2.4 Matplotlib

Matplotlib è una libreria per la creazione di grafici per il linguaggio di programmazione Python e la libreria matematica NumPy. Fornisce API orientate agli oggetti che permettono di inserire grafici all'interno di applicativi utilizzando toolkit GUI generici.
Tale libreria è stata utilizzata semplicemente per generare tutti i grafici presenti nel progetto (ed inseriti in questo elaborato).

Capitolo 3

Dataset

I dati presenti nel dataset (disponibile gratuitamente per l'uso pubblico[1]) sono stati raccolti durante 8 ore di attività, 2 ore con ciascuno dei 4 soggetti: due uomini e due donne, tutti adulti e sani (le caratteristiche di ciascun soggetto vengono mostrate in Figura 3.1). Ogni attività è stata svolta separatamente.

Subject	Genre	Age	Height	Weight	Instances
A	Female	46 y.o.	1.62m	67kg	51,577
B	Female	28 y.o.	1.58m	53kg	49,797
C	Male	31 y.o.	1.71m	83kg	51,098
D	Male	75 y.o.	1.67m	67kg	13,161*

* A smaller number of observed instances because of the participant's age

Figura 3.1: Caratteristiche dei partecipanti.

In totale sono stati raccolti 165.633 campioni per lo studio; la loro distribuzione tra le 5 classi è illustrata in Figura 3.2.

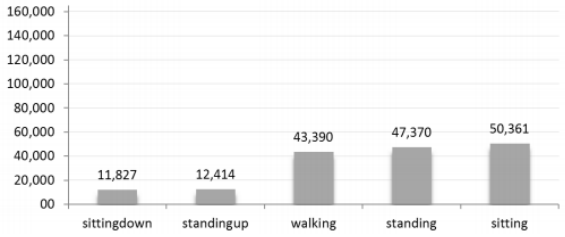

Figura 3.2: Frequenza delle classi tra i dati raccolti.

Nella Figura 3.3 viene mostrata invece la distribuzione degli utenti nelle classi.

[1] `http://groupware.les.inf.puc-rio.br/har`

Figura 3.3: Distribuzione degli utenti nelle classi.

Il formato utilizzato per il salvataggio dei dati è il *CSV* (*Comma-Separated Values*), formato largamente utilizzato per l'importazione ed esportazione dei dati in formato digitale.

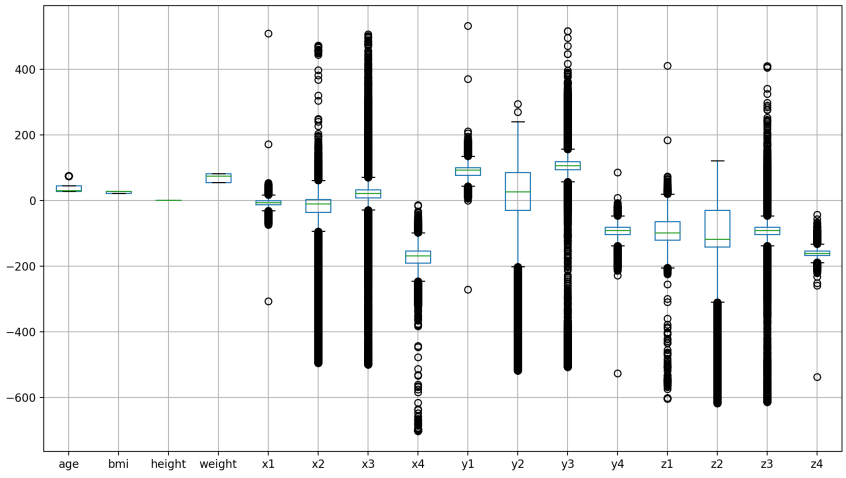

Figura 3.4: Box plot degli attributi del dataset iniziale.

Attributo	**Tipo**	**Descrizione**
User	String	Nome del soggetto
Gender	String	Sesso del soggetto
Age	Int	Età del soggetto
Height (cm)	Int	Altezza del soggetto
Weight (kg)	Int	Peso del soggetto
BMI	Float	Body Mass Index del soggetto
x1	Int	Valore letto sull'asse x del sensore sul bacino
y1	Float	Valore letto sull'asse y del sensore sul bacino
z1	Float	Valore letto sull'asse z del sensore sul bacino
x2	Float	Valore letto sull'asse x del sensore sulla coscia sinistra
y2	Float	Valore letto sull'asse y del sensore sulla coscia sinistra
z2	Float	Valore letto sull'asse z del sensore sulla coscia sinistra
x3	Float	Valore letto sull'asse x del sensore sulla caviglia destra
y3	Float	Valore letto sull'asse y del sensore sulla caviglia destra
z3	Float	Valore letto sull'asse z del sensore sula caviglia destra
x4	Float	Valore letto sull'asse x del sensore sul braccio destro
y4	Float	Valore letto sull'asse y del sensore sul braccio destro
z4	Float	Valore letto sull'asse z del sensore sul braccio destro
Classes	String	Classe predetta dai dati rilevati sui sensori

Tabella 3.1: Descrizione dataset iniziale.

Capitolo 4

Preprocessing

Il lavoro da noi svolto è stato eseguito prendendo come riferimento il lavoro di W. Ugolino *et. al* [1] dove vengono proposte una serie di operazioni di preprocessing sul dataset di riferimento.

Sono state realizzate due versioni di preprocessing:

- una versione "fedele" al paper di W. Ugolino *et. al*;
- una versione da noi creata che cerca di migliorare l'accuracy finale del modello.

4.1 Versione W. Ugolino *et. al*

Sui dati raccolti dagli accelerometri è stata eseguita una pre-elaborazione dei dati (*preprocessing*), seguendo alcune regole.
Il dataset è stato innanzitutto suddiviso in 20 partizioni (4 utenti per 5 classi differenti). Ogni partizione è stata ripartita in finestre di 8 misurazioni, corrispondenti a circa 1 secondo, con lo scopo di rappresentare nel miglior modo possibile, con l'aiuto della statistica descrittiva, i movimenti dell'utente. Le *features* derivanti dell'accelerazione sugli assi x, y, z sono le seguenti:

- per ogni accelerometro (una lettura): rotazione di angolo ψ attorno all'asse x (*roll*), rotazione di angolo θ attorno all'asse y (*pitch*) e modulo del vettore accelerazione;

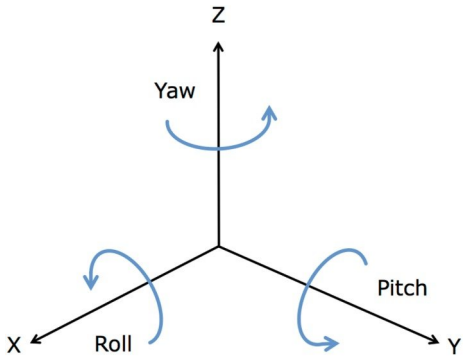

Figura 4.1: Roll, Pitch e Yaw.

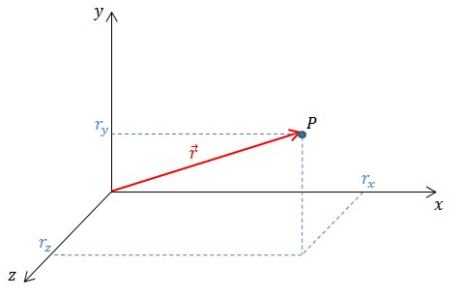

Figura 4.2: Modulo del vettore di accelerazione.

- per ogni finestra (circa 8 letture): varianza di roll, varianza di pitch e modulo dell'accelerazione per ogni accelerometro; inoltre vengono calcolate media e deviazione standard dei moduli d'accelerazione per i quattro accelerometri.

$$\bar{x} = \frac{x_1 + x_2 + \ldots + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Figura 4.3: Media matematica.

Sample Variance

$$s^2 = \frac{\sum(x - \bar{x})^2}{n - 1}$$

Sample Standard Deviation

$$s = \sqrt{\frac{\sum(x - \bar{x})^2}{n - 1}}$$

Figura 4.4: Varianza e deviazione standard.

L'attributo *classes* del dataset è stato discretizzato associandogli un valore intero, ciascuno corrispondente ad una classe, vedi figura 4.5).

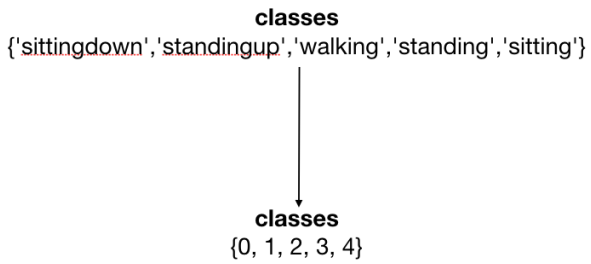

Figura 4.5: Discretizzazione dell'attributo classes.

Analisi delle features estratte

I seguenti grafici mostrano le distribuzioni dei dati (con valori continui) dopo il calcolo delle features.

Valori di varianza *roll*:

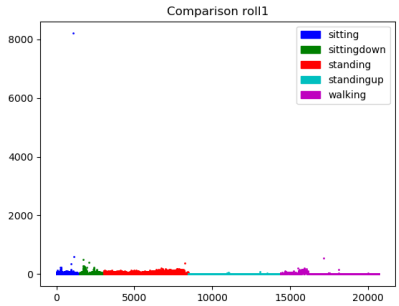

Figura 4.6: Roll1.

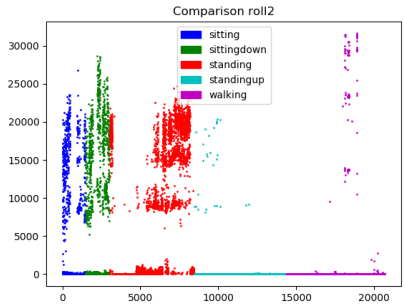

Figura 4.7: Roll2.

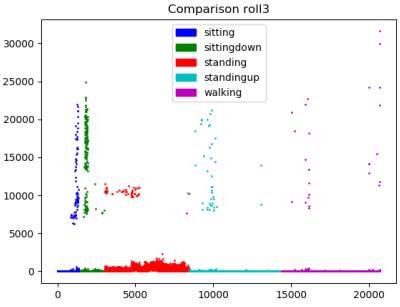

Figura 4.8: Roll3.

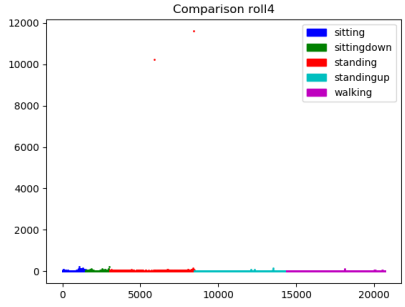

Figura 4.9: Roll4.

Valori di varianza *pitch*:

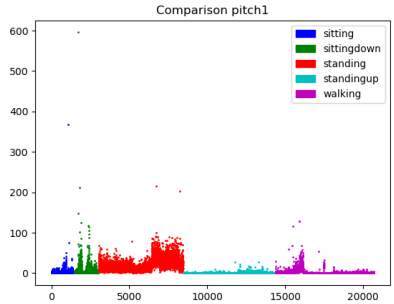

Figura 4.10: Pitch1.

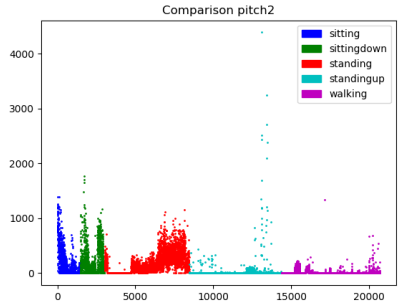

Figura 4.11: Pitch2.

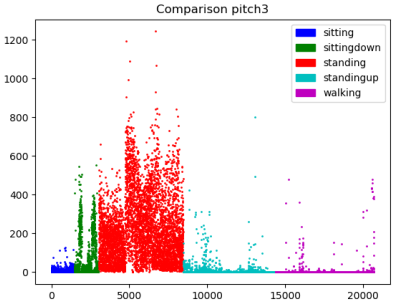

Figura 4.12: Pitch3.

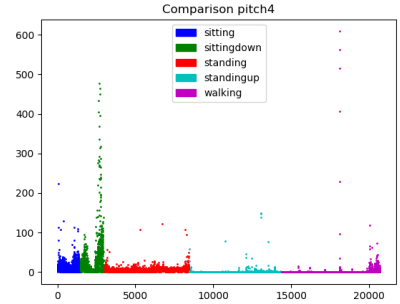

Figura 4.13: Pitch4.

Valori dei *moduli di accelerazione*:

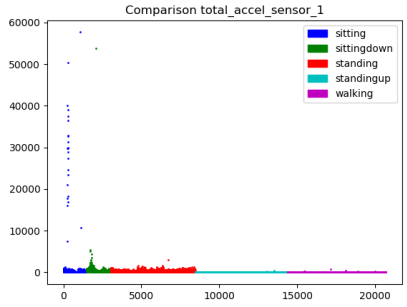

Figura 4.14: Modulo accelerazione 1.

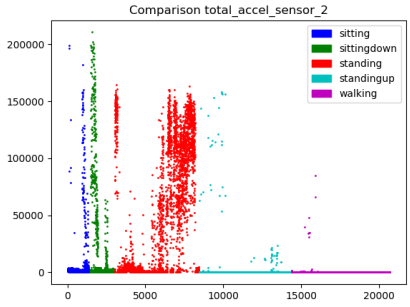

Figura 4.15: Modulo accelerazione 2.

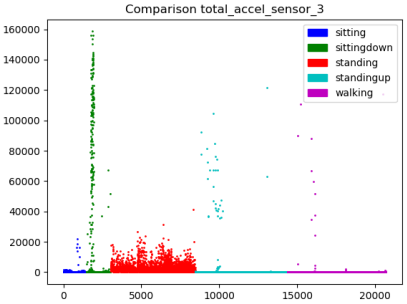

Figura 4.16: Modulo accelerazione 3.

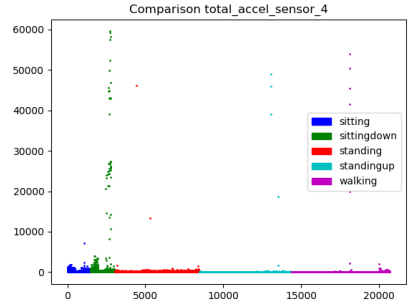

Figura 4.17: Modulo accelerazione 4.

Accelerazione media e relativa *deviazione standard*:

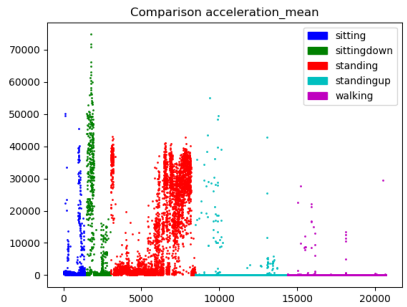

Figura 4.18: Accelerazione media.

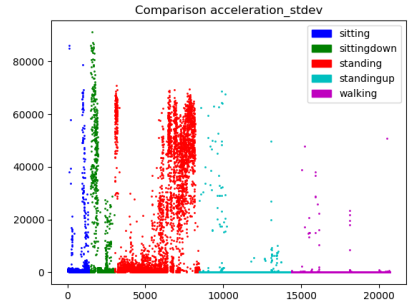

Figura 4.19: Deviazione standard dell'accelerazione.

Features selection

Secondo quanto riportato nel Paragrafo 4.2 dell'Articolo [1], per ciascun accelerometro sono state estratte le seguenti features:

- (1) Sensore sul bacino: modulo dell'accelerazione, varianza di pitch e varianza di roll;
- (2) Sensore sulla coscia sinistra: modulo dell'accelerazione e varianza di pitch;
- (3) Sensore sulla caviglia destra: varianza di pitch e varianza di roll;
- (4) Sensore sul braccio destro: modulo dell'accelerazione;
- Per tutti i sensori: accelerazione media e deviazione standard dell'accelerazione.

  Gli altri parametri sono stati eliminati.

Normalizzazione e discretizzazione

Dopo aver svolto l'operazione di features selection, i dati del dataset vengono sottoposti a due nuove trasformazioni, che hanno l'obiettivo di gestire in maniera più efficiente la distribuzione dei dati e di facilitare la creazione delle reti Bayesiane.

La prima operazione applicata è la normalizzazione, usata per ridimensionare l'intervallo dei dati in [0,1].

$$\tilde{x}_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

Figura 4.20: Formula di normalizzazione utilizzata.

Successivamente, è stato svolto un lavoro di discretizzazione, con lo scopo di ottenere valori interi per rendere più semplice la creazione dei modelli Bayesiani. L'intervallo di discretizzazione scelto è [0:10] (valori compresi tra 0 e 10, estremi inclusi), in quanto a seguito di varie sperimentazioni, risulta essere il valore con il quale i dati vengono distribuiti nella maniera più ottimale.
L'operazione di discretizzazione è stata svolta con l'utilizzo della funzione *cut*, presente nella libreria *pandas*.

pandas.**cut**(*x*, *bins*, *right=True*, *labels=None*, *retbins=False*, *precision=3*, *include_lowest=False*, *duplicates='raise'*)
[source]

Bin values into discrete intervals.

Figura 4.21: Funzione *cut* della libreria *pandas*.

Dataset finale

A seguito delle operazioni di preprocessing, il dataset risulta così strutturato:

Attributo	**Tipo**
acceleration_mean	Int
acceleration_stdev	Int
pitch1	Int
pitch2	Int
pitch3	Int
roll1	Int
roll2	Int
roll3	Int
total_accel_sensor_1	Int
total_accel_sensor_2	Int
total_accel_sensor_4	Int
classes	Int

Tabella 4.1: Descrizione attributi del dataset finale.

Il dataset finale è composto da 20.695 record, rispetto ai 165.633 dei campioni raccolti per lo studio. Il numero è inferiore in quanto i dati sono stati suddivisi in finestre di 1 secondo, equivalenti a circa 8 misurazioni.

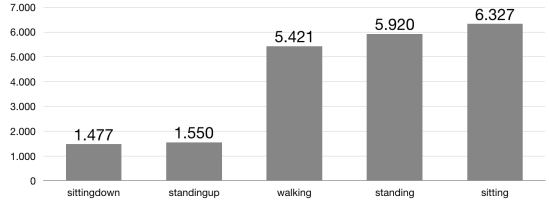

Figura 4.22: Frequenza delle classi del dataset finale.

Nella seguente Figura 4.23 viene riportata la matrice di correlazione di Pearson:

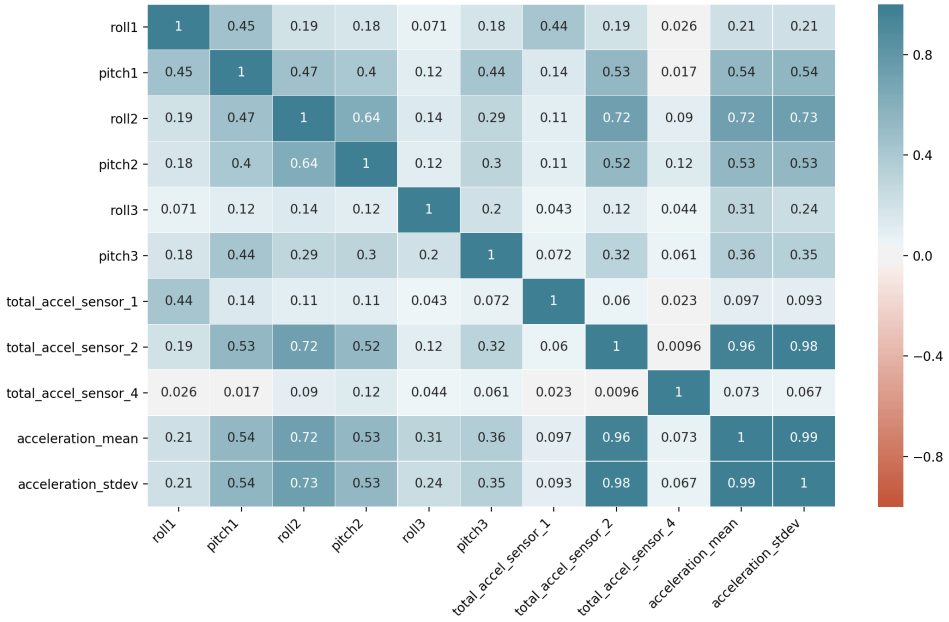

Figura 4.23: Matrice di correlazione di Pearson.

Train e Test

I dati presenti nel dataset finale (20.695 record) sono stati suddivisi in *Train* e *Test* come segue:

- *Train*: 16.556 record, pari all'80% dei record contenuti nel dataset;
- *Test*: 4.139 record, pari al 20% dei record contenuti nel dataset.

I record contenuti in Train sono stati utilizzati come dati iniziali per apprendere la struttura della rete Bayesiana, mentre i record di Test sono stati utilizzati successivamente come valori da predirre una volta appresa la struttura della rete.

4.2 La ”nostra” versione

Di seguito viene presentata la nostra implementazione; essa tiene sempre in considerazione l'Articolo [1], ma vengono effettuate delle lievi modifiche con l'obiettivo di migliorare le performance di riconoscimento.

Analizzando i dati del dataset iniziale e i valori delle coordinate xyz dei vari accelerometri, sono stati rilevati alcuni record con valori che non rispettano la distribuzione generale (anomalia probabilmente dovuta ad errori di misura degli accelerometri).

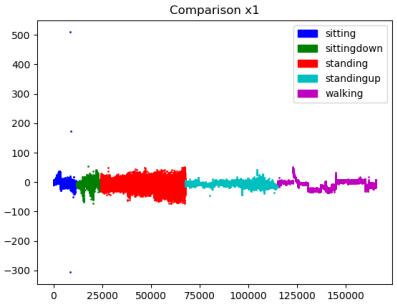

Figura 4.24: Distribuzione dei valori di x1.

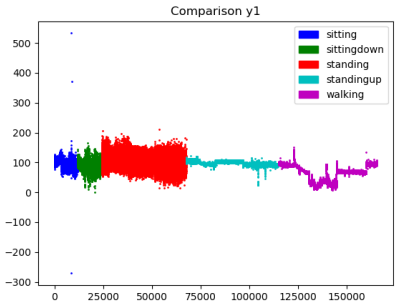

Figura 4.25: Distribuzione dei valori di y1.

Per esempio, visionando le Figure 4.24 e 4.25 (corrispondenti ai valori di x1 e y1) è possibile notare come alcuni punti (record) siano identificabili come *outliers*. Al fine di ridurre il rumore nei dati tali record sono stati rimossi. Il numero di campioni osservati è passato quindi da 165.633 a 165.376, con 257 record rimossi. In questo modo la distribuzione dei dati risulta essere più omogenea.

Le seguenti Figure 4.26, 4.27 e 4.28 presentano alcuni esempi che mostrano la distribuzione dei dati prima e dopo l'operazione di rimozione.

Figura 4.26: Distribuzione dei valori di x1, prima (sinistra) e dopo (destra).

Figura 4.27: Distribuzione dei valori di y3, prima (sinistra) e dopo (destra).

Figura 4.28: Distribuzione dei valori di z4, prima (sinistra) e dopo (destra).

Features

Analogamente al preprocessing visto nel Paragrafo 4.1, per ogni record sono stati calcolati i valori di roll, pitch e dei moduli dei vettori di accelerazione.

Per ogni finestra (circa 8 letture per secondo), invece di calcolare la varianza delle features estratte, ne è stata calcolata la media.
Questo perchè, a seguito di varie sperimentazioni (trasformazioni matematiche e relativa analisi dei dati), è stato notato che la media denota una distinzione più marcata tra le classi.

Di seguito (Figure 4.29 e 4.30) vengono presentati una serie di esempi che mostrano la differenza di correlazione di varianza e media rispetto alle classi:

Figura 4.29: Accelerometro sul bacino. Confronto pitch1, varianza (sinistra) e media (destra).

Figura 4.30: Accelerometro sul braccio. Confronto roll4, varianza (sinistra) e media (destra).

Analisi delle features

I seguenti grafici mostrano le distribuzioni dei dati dopo aver calcolato la media delle features per ogni finestra (circa 8 letture).

Valori di *roll*:

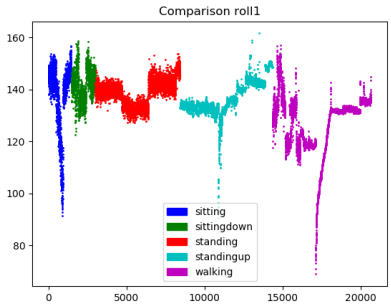

Figura 4.31: Roll1.

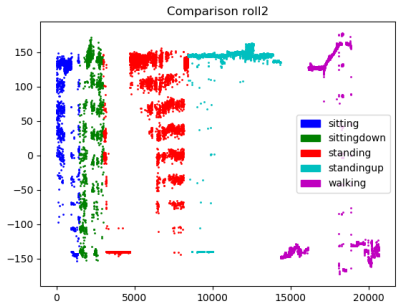

Figura 4.32: Roll2.

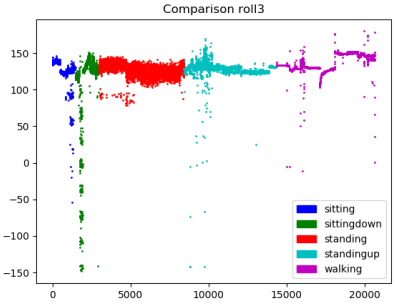

Figura 4.33: Roll3.

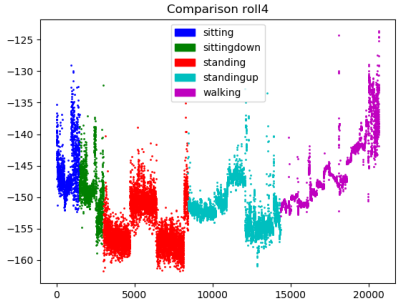

Figura 4.34: Roll4.

Valori di *pitch*:

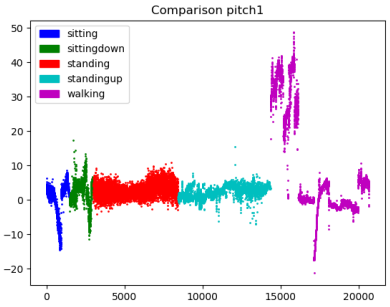

Figura 4.35: Pitch1.

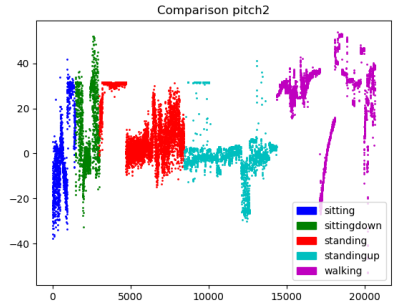

Figura 4.36: Pitch2.

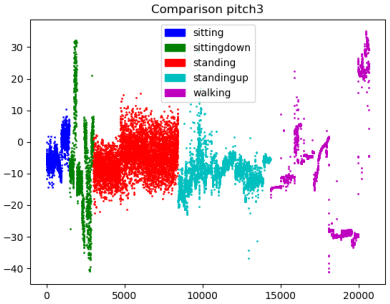

Figura 4.37: Pitch3.

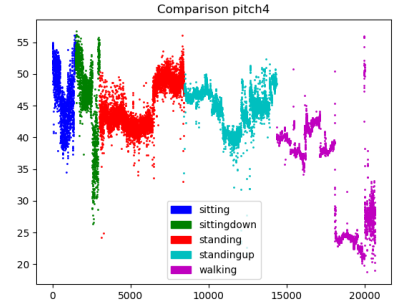

Figura 4.38: Pitch4.

Valori dei *moduli di accelerazione*:

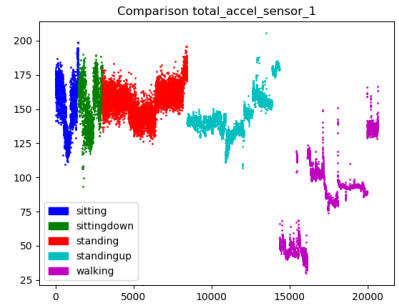

Figura 4.39: Modulo accelerazione 1.

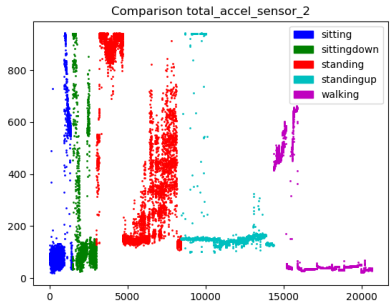

Figura 4.40: Modulo accelerazione 2.

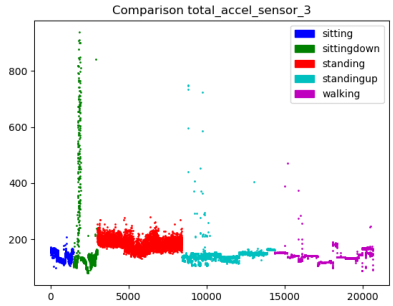

Figura 4.41: Modulo accelerazione 3.

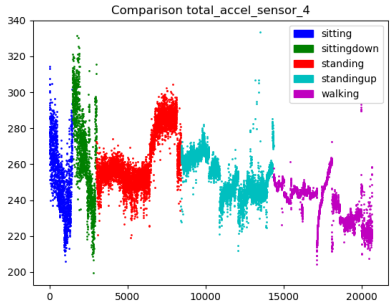

Figura 4.42: Modulo accelerazione 4.

Accelerazione media e relativa *deviazione standard*:

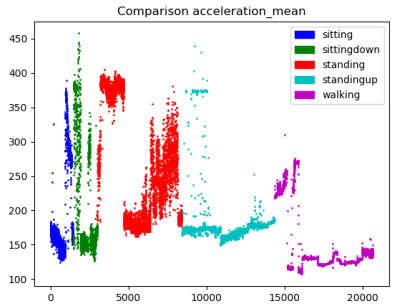

Figura 4.43: Accelerazione media.

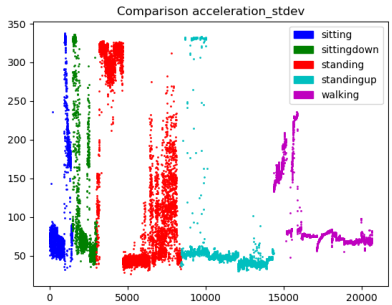

Figura 4.44: Deviazione standard dell'accelerazione.

Dataset finale

A seguito delle operazioni di preprocessing, il numero di record risulta essere 20.665 (rispetto ai 20.695 risultanti dalla fase di preprocessing vista precedentemente nella Sezione 4.1).

Le operazioni di features extraction, normalizzazione e discretizzazione rimangono le stesse dell'implementazione precedente (Sezione 4.1).

Gli attributi del dataset finale sono quelli riportati nella Tabella 4.1. Si riporta nella seguente Figura 4.45 la matrice di correlazione di Pearson:

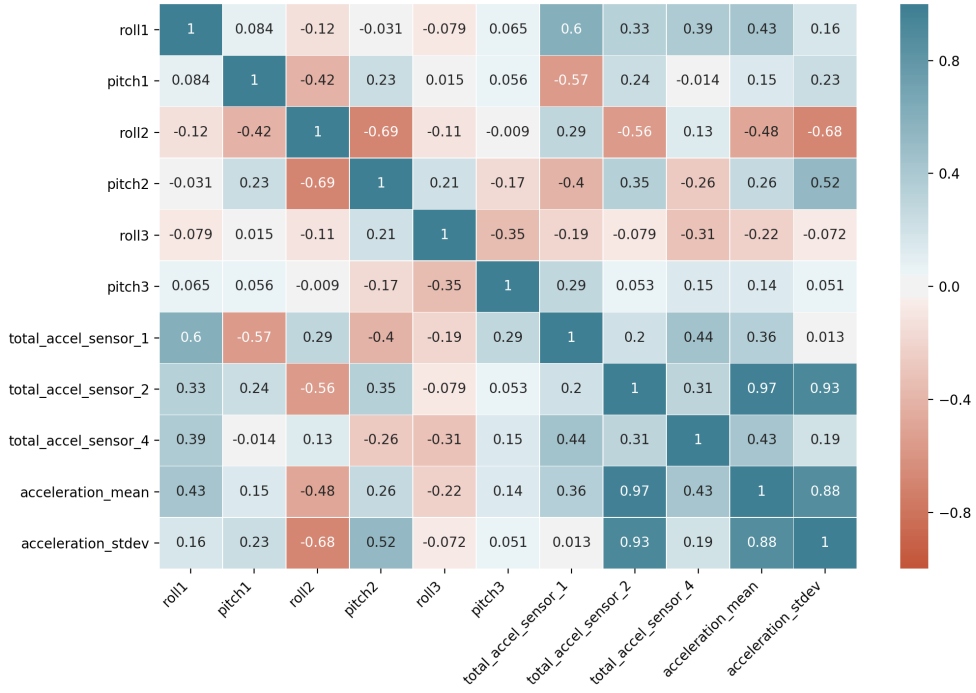

Figura 4.45: Matrice di correlazione di Pearson.

Train e Test

I dati presenti nel dataset finale (20.665 record) sono stati suddivisi in *Train* e *Test* come segue:

- *Train*: 16.532 record, pari all'80% dei record contenuti nel dataset;
- *Test*: 4.133 record, pari al 20% dei record contenuti nel dataset.

Capitolo 5

Reti Bayesiane

Una rete Bayesiana (*Bayesian network*) è un modello grafico probabilistico che rappresenta un insieme di variabili stocastiche con le loro dipendenze condizionali attraverso l'uso di un grafo aciclico diretto (DAG). Formalmente dunque, le reti Bayesiane sono grafi diretti aciclici i cui nodi rappresentano variabili casuali in senso Bayesiano: possono essere quantità osservabili, variabili latenti, parametri sconosciuti o ipotesi. Gli archi rappresentano condizioni di dipendenza; i nodi che non sono connessi rappresentano variabili che sono condizionalmente indipendenti tra loro. Ad ogni nodo è associata una funzione di probabilità che prende in input un particolare insieme di valori per le variabili del nodo genitore e restituisce la probabilità della variabile rappresentata dal nodo. Per esempio, se i genitori del nodo sono variabili booleane allora la funzione di probabilità può essere rappresentata da una tabella in cui ogni entry rappresenta una possibile combinazione di valori vero o falso che i suoi genitori possono assumere. Esistono algoritmi efficienti che effettuano inferenza e apprendimento a partire dalle reti Bayesiane.

5.1 Apprendere strutture di reti Bayesiane

L'approccio più semplice per apprendere la struttura di una rete Bayesiana partendo dai dati è eseguire una *ricerca*: partendo da un modello che non contiene alcun collegamento, si possono aggiungere i genitori ad ogni nodo, calibrando i parametri con i metodi di apprendimento statistico e misurando l'accuratezza del modello risultante. Alternativamente, è possibile partire da una stima iniziale della struttura e utilizzare metodi come *hill-climbing* e *simulated annealing* per modificarla, ricalcolando i parametri ogni volta. Le modifiche possono includere l'inversione, l'aggiunta o la cancellazione di archi. Il processo non deve introdurre cicli, perciò molti algoritmi impongono un ordimanento sulle variabili e permettono che un nodo abbia genitori solo tra i nodi che lo precedono nell'ordinamento.

5.2 Esperimenti

Durante la fase di sperimentazione sono stati creati diversi modelli, in particolar modo sono stati utilizzati algoritmi che si basano sui concetti di apprendimento spiegati nel precedente Paragrafo 5.1, implementati con le librerie *PGMPY* e *Pomegranate*.

5.2.1 PGMPY

Di seguito viene presentato il funzionamento dell'algoritmo *Hill-Climbing* seguito dai modelli che sono stati generati con la libreria *PGMPY*.

Hill-Climbing

Hill-Climbing è l'implementazione più semplice di un algoritmo genetico[1]. Elimina completamente concetti come *population* e *crossover*, concentrandosi invece sulla facilità di implementazione. Ha iterazioni più veloci rispetto agli algoritmi genetici tradizionali, ma è meno approfondito di questi ultimi.

Il funzionamento di Hill-Climbing è molto semplice, e può essere riassunto nei seguenti passi:

1. Inizia con una soluzione vuota o casuale (detta *best solution*);
2. Si fa una copia della soluzione, dopodiché la si muta leggermente;
3. Valutazione della nuova soluzione: se è migliore della *best solution*, quest'ultima viene sostituita con la nuova soluzione;
4. Si ritorna al punto 2. e si ripete il procedimento.

L'algoritmo termina nel momento in cui raggiunge il punto massimo, tale per cui nessun punto vicino ha valore maggiore.

[1] `https://it.wikipedia.org/wiki/Algoritmo_genetico`

Modelli

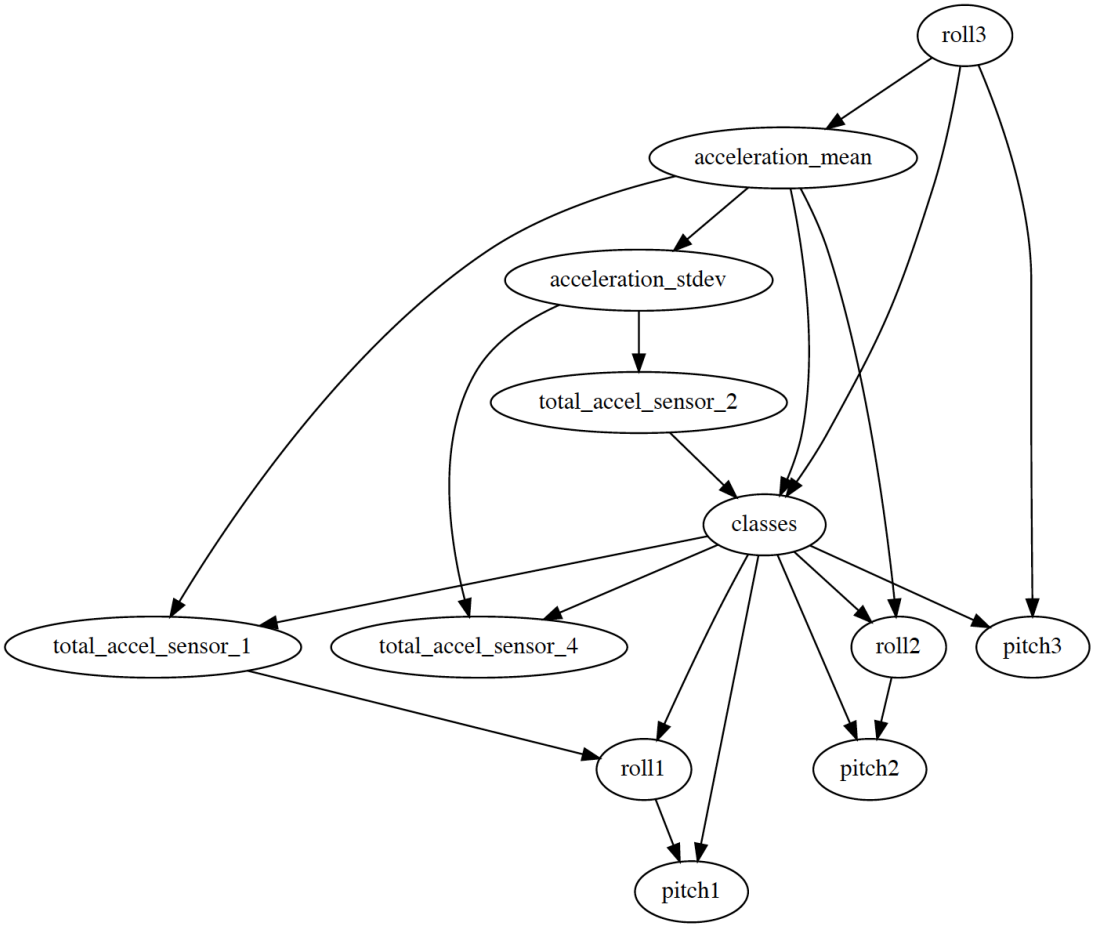

Figura 5.1: Rete generata a partire dai dati preprocessati come da [1] (algoritmo Hill-Climbing)

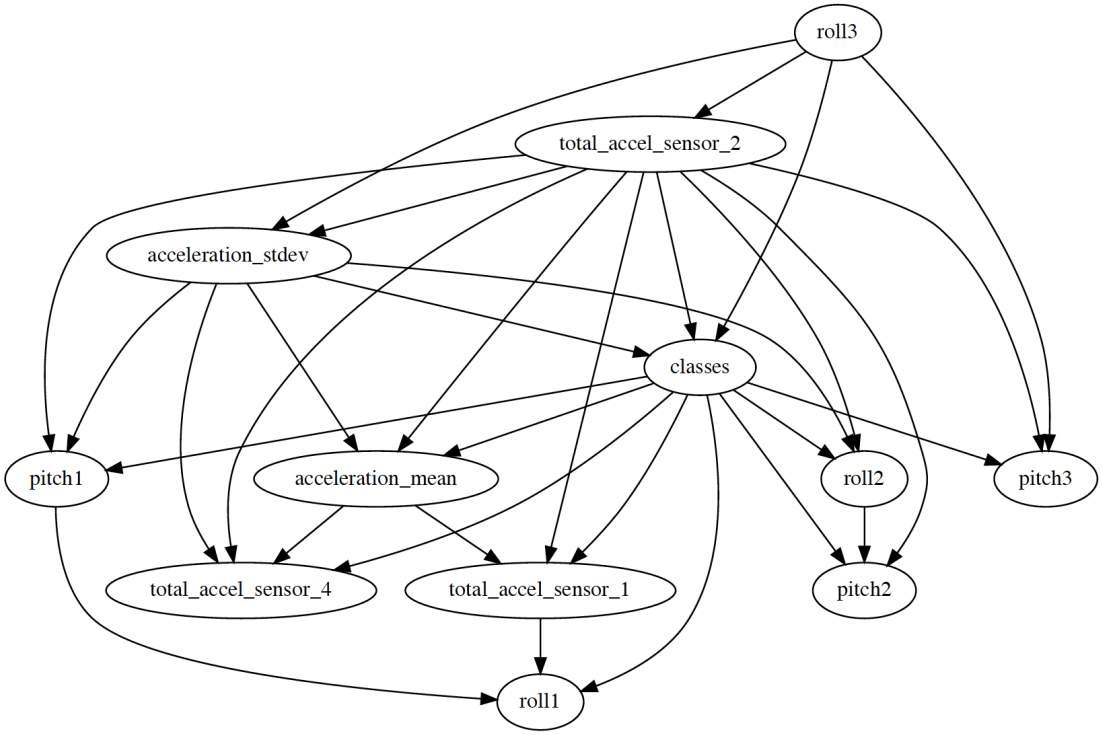

Figura 5.2: Rete generata a partire dai dati preprocessati con le nostre modifiche (algoritmo Hill-Climbing)

5.2.2 Pomegranate

Analogamente al Paragrafo 5.2.1 vengono presentati l'algoritmo utilizzato da Pomegranate e le reti create con esso.

Algoritmo greedy per la ricerca

Un algoritmo *greedy* per la ricerca è un algoritmo di apprendimento della struttura derivante dall'algoritmo DP/A (programmazione dinamica). Sostanzialmente, l'algoritmo DP/A considera dinamicamente qualsiasi ordinamento topologico delle variabili e sceglie il migliore set di *"parents"* per ognuna di esse; in modo tale che l'ordinamento risulti come il migliore abbinato al set di padri per ogni variabile dato quell'ordinamento.
Invece l'algoritmo greedy da noi utilizzato costruisce l'ordinamento topologico in maniera greedy, ovvero sceglie la variabile successiva migliore da aggiungere all'ordinamento già esistente ad ogni step. Trovata la struttura migliore, i parametri vengono stimati utilizzando la Maximum Likelihood Estimation.

Modelli

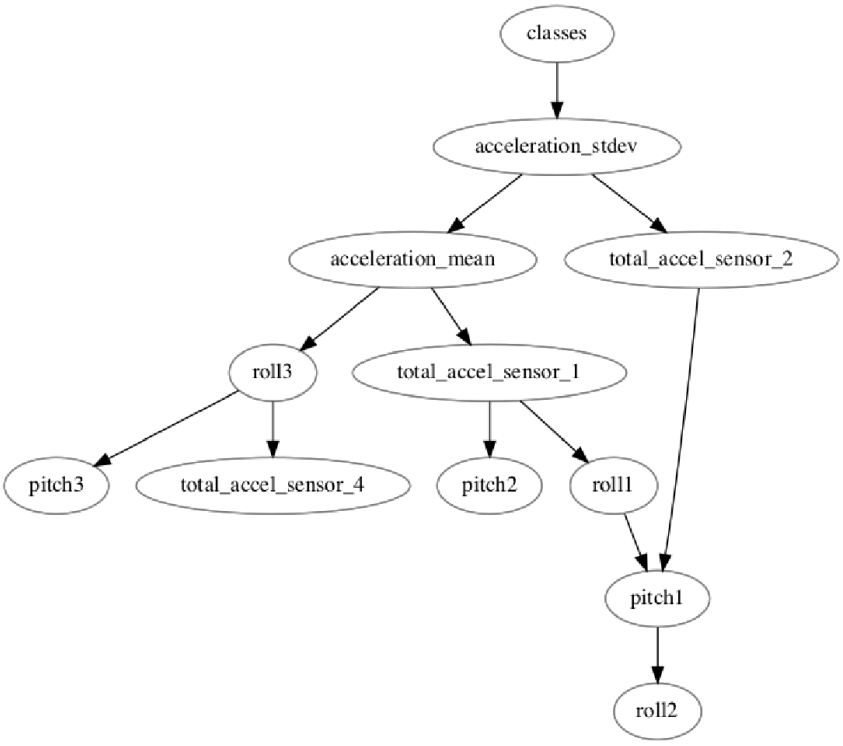

Figura 5.3: Rete generata a partire dai dati preprocessati come da [1] (algoritmo greedy)

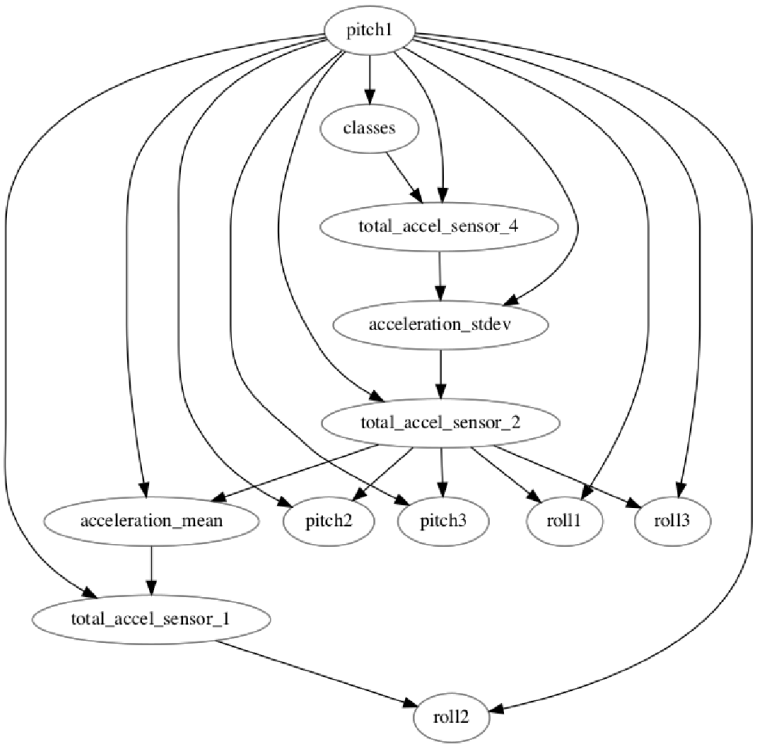

Figura 5.4: Rete generata a partire dai dati preprocessati con le nostre modifiche (algoritmo greedy)

Capitolo 6

Analisi dei risultati

In questo Capitolo vengono riportati i risultati ottenuti con i modelli presentati nel Paragrafo 5.2.

La Tabella 6.1 mostra una overview dei risultati ottenuti. Le metriche utilizzate per la valutazione sono *accuracy*, *precision*, *recall* e *f1-score*:

		Accuracy	***Precision***	***Recall***	***F1-Score***
Ugolino Version	*PGMPY*	**0.68%**	**0.68%**	**0.62%**	**0.62%**
	POMEGRANATE	0.53%	0.38%	0.41%	0.37%
Our Version	*PGMPY*	0.89%	0.86%	0.85%	0.82%
	POMEGRANATE	**0.94%**	**0.89%**	**0.91%**	**0.90%**

Tabella 6.1: Risultati ottenuti.

Di seguito vengono mostrate le matrici di confusione generate dai modelli con accuracy migliore e le metriche relative ad ogni classe.

Versione di W.Ugolino *et al.*

Analizzando la precedente Tabella 6.1, la rete migliore seguendo il preprocessing di W. Ugolino *et al.* [1] è quella generata utilizzando l'hill-climbing di *PGMPY*: 0.68%.

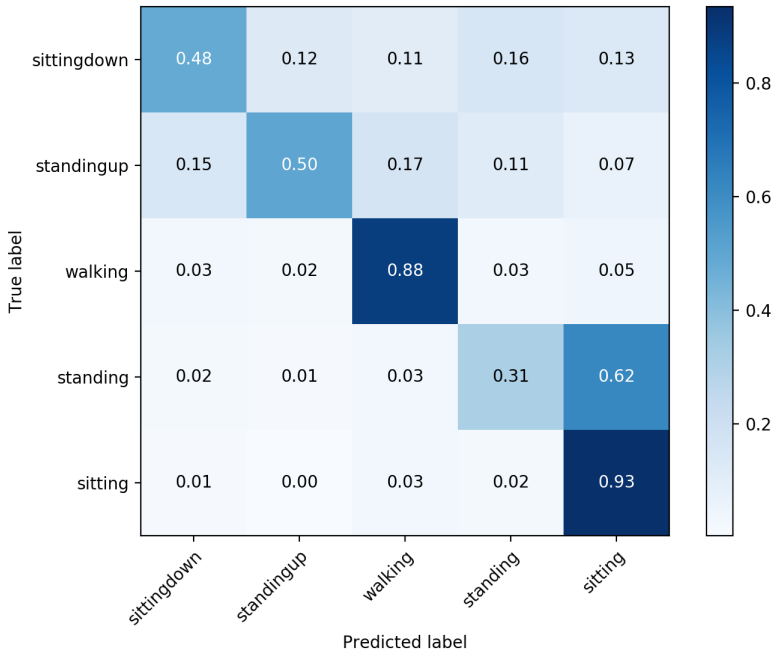

Figura 6.1: Matrice di confusione del modello creato a partire dal preprocessing illustrato nel Paragrafo 4.1 (Ugolino *et al.* [1]) (Libreria *PGMPY*).

	Sitting Down	***Standing Up***	***Walking***	***Standing***	***Sitting***
F1-Score	0.51	0.58	**0.87**	0.43	0.71
Precision	0.54	0.67	**0.86**	0.73	0.58
Recall	0.48	0.50	0.88	0.31	**0.93**

Tabella 6.2: Metriche di valutazione performance suddivise per classe.

La rete Bayesiana è in grado di prevedere l'attività svolta da parte dei pazienti con un'accuracy del 68%. In particolare si può dedurre che il modello riconosce in maniera ottimale i movimenti di *walking* e *sitting* (rispettivamente 88% e 93%). Situazione contraria per quanto riguarda *standing*, che viene riconosciuta come *sitting* nel 62%

dei casi; questa imprecisione è denotata anche dal suo valore di recall, ovvero quante volte una sua istanza viene predetta come *standing*, quando effettivamente lo è (ed il valore è basso). Inoltre *sittingdown* e *standingup* hanno una percentuale di veri positivi più bassa in quanto nel dataset c'è una quantità minore di campioni rispetto alle altre classi.

La "nostra" versione

Il modello migliore generato dal preprocessing visto precedentemente nel Paragrafo 4.2 risulta quello generato utilizzando l'algoritmo greedy di *Pomegranate* (che restituisce un accuracy pari al 0.94%). Si riporta nella Figura 6.2 la matrice di confusione relativa alle performance di tale modello sul *Test*.

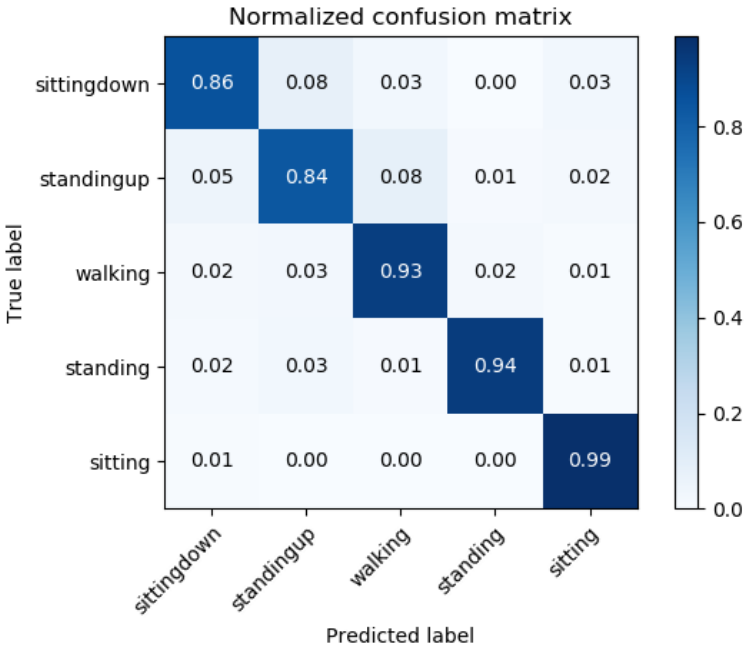

Figura 6.2: Matrice di confusione del modello creato a partire dal preprocessing visto nel Paragrafo 4.2 ("nostra" versione) (Libreria *Pomegranate*).

	Sitting Down	***Standing Up***	***Walking***	***Standing***	***Sitting***
F1-Score	0.83	0.78	0.94	0.96	**0.98**
Precision	0.80	0.74	0.95	**0.98**	**0.98**
Recall	0.86	0.84	0.93	0.94	**0.98**

Tabella 6.3: Metriche di valutazione performance suddivise per classe.

Analizzando il valore di accuracy e la matrice di confusione si può affermare che il modello è in grado di prevedere in maniera quasi ottimale le attività dei pazienti.

Anche qui le performance del modello nel riconoscere le attività di *sittingdown* e *standingup* risultano essere inferiori a causa della più bassa frequenza dei campioni nel dataset.

Capitolo 7

Interfaccia grafica

E' possibile sperimentare le performance della rete bayesiana attraverso un'applicazione web, realizzata con *Flask* (microframework per *Python*).
Non avendo a disposizione gli accelerometri usati per misurare i valori delle coordinate di una determinata posizione, l'applicazione permette all'utente di selezionare in maniera casuale delle misurazioni presenti all'interno del dataset di *Test*.

Il modello utilizzato per predire i movimenti è già stato presentato nella Figura 5.4, ovvero la rete Bayesiana modellata con Pomegranate sul dataset preprocessato secondo la "nostra" versione (Paragrafo 4.2).

Per ogni predizione, oltre a riportare la classe predetta, vengono riportate anche le probabilità calcolate per ognuno dei possibili movimenti.

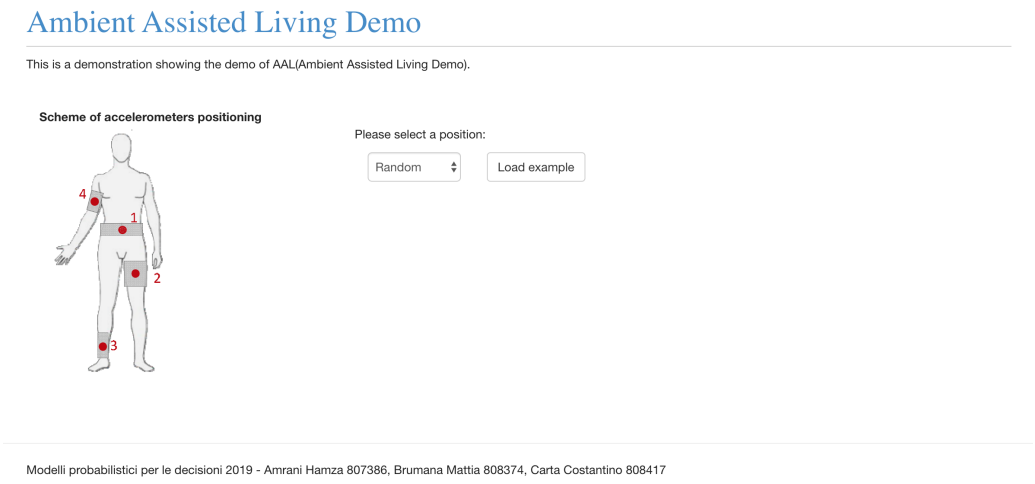

Figura 7.1: Schermata Home.

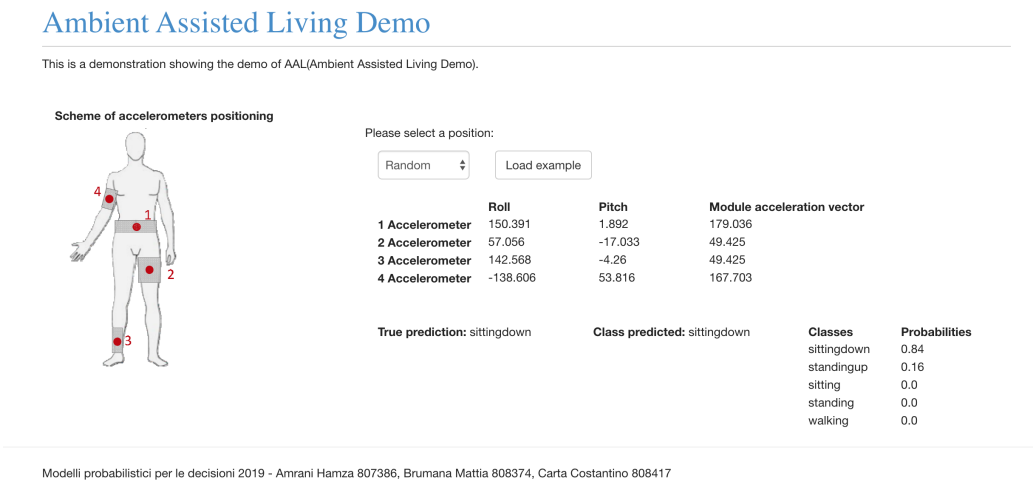

Figura 7.2: Esempio di utilizzo dell'applicazione. In basso a destra è possibile vedere le probabilità ottenute per ogni classe.

Capitolo 8

Conclusioni

L'obiettivo di questo progetto è stato quello di creare un modello di apprendimento in grado di riconoscere automaticamente il movimento di un soggetto date le rilevazioni di quattro accelerometri posti sul proprio corpo.

La soluzione proposta al problema comporta l'utilizzo di reti Bayesiane, create con i dati ricavati dal dataset a noi fornito (Capitolo 3), dopo aver effettuato delle operazione di preprocessing viste nel Capitolo 4.
In particolare, relativamente al preprocessing, sono stati presentati due approcci (Paragrafi 4.1 e 4.2).

I risultati degli esperimenti, presentati nel Paragrafo 5.2, hanno mostrato come l'utilizzo di reti Bayesiane porti ad ottime prestazioni in termini di accuracy. Le migliori percentuali di riconoscimento risultano essere del 68% per quanto riguarda la rete creata dai dati preprocessati secondo W.Ugolino *et. al* [1] e del 94% per una rete generata a partire dai dati preprocessati secondo la "nostra" versione (6).

Per sperimentare il riconoscimento dei movimenti è stata realizzata un'applicazione web in Flask (Capitolo 7).

Sviluppi futuri

E' importante tenere in considerazione che i dati con il quale sono state apprese le reti Bayesiane derivano dalle misurazioni effettuate su quattro soggetti. Un possibile sviluppo futuro potrebbe essere quello di ampliare il dataset con rilevazioni effettuate su un maggior numero di persone, con età, peso, altezza e sesso differenti, con l'obiettivo di ampliare la conoscenza del modello relativa ai movimenti (che possono differire di molto per caratteristiche fisiche).

Bibliografia

[1] K. Vega E. Velloso R. Milidiù H. Fuks W. Ugolino, D. Cardador. Wearabale computing: Accelerometers' data classification of body postures and movements. 2012.