Università Politecnica delle Marche
Facoltà di Ingegneria

Corso di Laurea Triennale in Ingegneria Elettronica

Tecniche di machine learning e data fusion applicate alla classificazione di attività fisiche tramite l'elaborazione di segnali fotopletismografici ed inerziali.

Machine learning and data fusion techinques applied to physical activities classification using photoplethysmographic and accelerometric signals

Relatore:
Prof. Claudio Turchetti
Correlatore:
Prof.ssa Laura Falaschetti

Candidato:
Edoardo Focante

Anno Accademico 2018/2019

Indice

Capitolo 1

Introduzione

La tesi è stata realizzata in collaborazione con il Dipartimento di Ingegneria della Informazione dell' Università Politecnica delle Marche, sotto la supervisione del prof. Claudio Turchetti e della dottoressa Laura Falaschetti. L'obiettivo della tesi è quello di verificare, attraverso la sperimentazione, la possibilità di classificare diverse attività fisiche mediante algoritmi noti di machine learning partendo da segnali fotopletismiografici e accelerometrici. Per fare ciò, si è ripreso il lavoro svolto da Leonardo Saraceni nella sua tesi [4], che è stato ampliato attraverso l'aggiunta di nuove sperimentazioni. Sono state infatti misurate le performance di diversi algoritmi di classificazione basati su machine learning, applicati ai dati in questione. Nei diversi casi di studio si è partito dallo stesso insieme di dati, ma manipolato in maniera leggermente diversa, al fine di ottenere i migliori risultati nella classificazione.

L'importanza del lavoro svolto deriva dalla sempre crescente diffusione di sensori integrati nei dispositivi di utilizzo quotidiano, come smartphone e smartwatch. Questi mezzi rendono possibile l'acquisizione di una grande quantità di dati che possono essere processati in tempo reale per monitorare lo stato fisico di un individuo e aiutarlo nelle attività di tutti i giorni. Questo ha una grande importanza nel campo medico e nel campo sportivo, anche se, attraverso le tecnologie sempre più accessibili, monitorare i propri parametri è diventato possibile per tutti, ad esempio per chi fa sport a livello amatoriale o per chi vuole semplicemente controllare la qualità del proprio sonno. Per fare questo molto spesso vengono utilizzate tecniche di machine learning, un'applicazione dell'intelligenza artificiale volta a realizzare applicazioni e dispositivi in grado di imparare ed agire in maniere simile all'uomo, migliorando le proprie prestazioni attraverso l'acquisizione di nuovi dati e l'interazione col mondo esterno. Questa disciplina è molto utile ed efficace in tutte le situazione in cui è necessario analizzare grandi moli di dati e trovare dei pattern al loro interno, motivo per il quale è risultata molto adatta al lavoro svolto nell'ambito di questa tesi.

Capitolo 2

Machine Learning

Il machine learning è una disciplina volta ad insegnare alle macchine ad agire come persone pensanti, in particolare ad imparare dalle proprie esperienze. Gli algoritmi di machine learning, infatti, utilizzano metodi per apprendere direttamente dai dati senza bisogno di un'equazione che gli imponga come agire. Per questo motivo il machine learning viene utilizzato per risolvere problemi complessi, con molti dati e molte variabili in gioco, che non possono essere risolti mediante una formula univoca. È infatti utilizzato per applicazioni come i filtri anti-spam, i sistemi di riconoscimento facciale, per le diagnosi mediche automatizzate, per la guida autonoma e molto altro. La disciplina del machine learning si divide in due grandi settori:

- Apprendimento supervisionato
- Apprendimento non supervisionato

2.1 Apprendimento supervisionato

L'apprendimento supervisionato punta a creare un modello partendo da una serie di input e una serie nota di output corrispondenti. Utilizzando questi, il modello viene allenato per poter prevedere i risultati di nuovi ingressi non noti, riuscendo a simulare così processi non descrivibili mediante un'equazione univoca oppure troppo complicati per essere trattati in maniera deterministica. Se la risposta da predire è discreta vengono utilizzate tecniche dette di classificazione, con lo scopo di dividere i dati in ingresso in diverse classi distinte. I dati utilizzati per l'apprendimento sono in questo caso provvisti di un'etichetta, che ne identifica la classe di appartenenza. La tecniche di classificazione sono utilizzate per esempio per capire se un'e-mail è spam o meno, se un segnale vocale è stato prodotto da un uomo o da una donna, o se una foglia è malata oppure sana. Se al contrario la risposta da predire è di tipo continuo, allora si parla di algoritmi di regressione. Ad esempio vengono utilizzati per prevedere variazioni delle quote borsistiche o dell'utilizzo energetico di case e città, oppure per effettuare previsioni metereologiche.

2.2 Apprendimento non supervisionato

L'apprendimento non supervisionato mira a riorganizzare un set di dati sprovvisti di etichette e a trovare caratteristiche comuni tra gli elementi che lo costituiscono. La

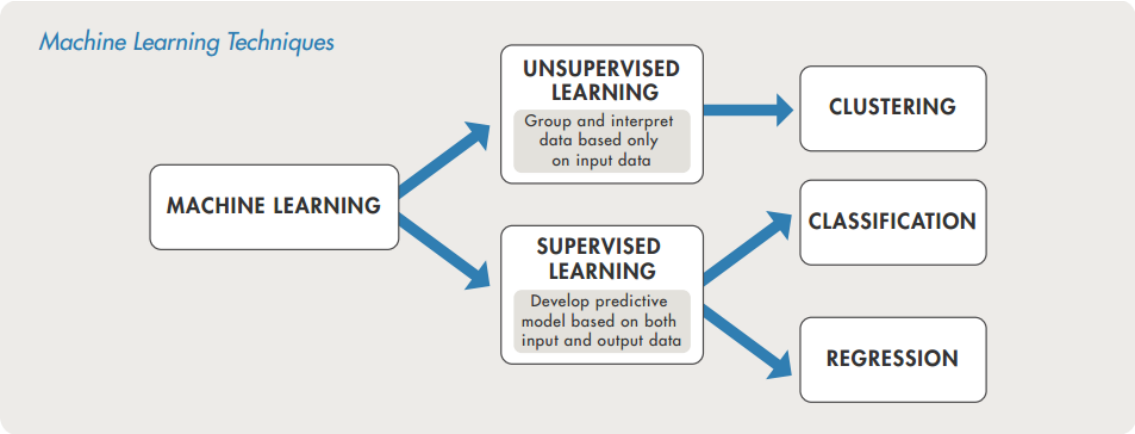


Figura 2.1: Classificazione degli algoritmi di machine learning

tecnica più diffusa di apprendimento non supervisionato è il clustering, che punta a suddividere i dati in ingresso in gruppi detti cluster, basandosi su caratteristiche comuni e similitudini. Gli elementi di uno stesso cluster saranno quindi simili tra loro e diversi dagli elementi appartenenti a cluster diversi. Esistono due tipi di clustering, il primo è detto hard clustering e prevede che ogni elemento del set di dati di partenza venga assegnato ad un singolo cluster. Il secondo, invece, è detto soft clustering e prevede che ogni elemento del set di dati possa appartenere ad uno o più cluster allo stesso tempo.

2.3 Algoritmi di Machine Learning

Esistono diversi algoritmi per creare un modello di classificazione. Non ne esiste, infatti, uno sempre migliore degli altri. Bisogna basare la valutazione della bontà di un algoritmo su diversi parametri quali la complessità del modello, l'utilizzo di memoria, l'interpretabilità e la precisione nella classificazione. In questa sezione ne verranno analizzati alcuni, in particolare gli algoritmi di apprendimento supervisionato utilizzati durante la sperimentazione. Nella trattazione si suppone che i dati utilizzati siano organizzati in una matrice detta data matrix, dove ad ogni riga corrisponde un diverso dato e ad ogni colonna un parametro differente di quel dato, detto caratteristica (feature). Nell'ambito dell'apprendimento supervisionato una delle colonne della data matrix, solitamente l'ultima, contiene le classi di appartenenza dei diversi dati.

2.3.1 Algoritmi di classificazione

Decision Tree

Nella scienza computazionale, il Decision tree learning usa un albero di decisione come un modello predittivo per andare da osservazioni su un oggetto (rappresentate sui rami), fino a conclusioni riguardo il valore dell'obbiettivo (rappresentato sulle foglie). È uno degli approcci mediante modelli predittivi più usato in statistica, data mining e machine learning. I modelli ad albero dove le variabili possono assumere solo un insieme di valori discreti, sono chiamati alberi di classificazione e hanno

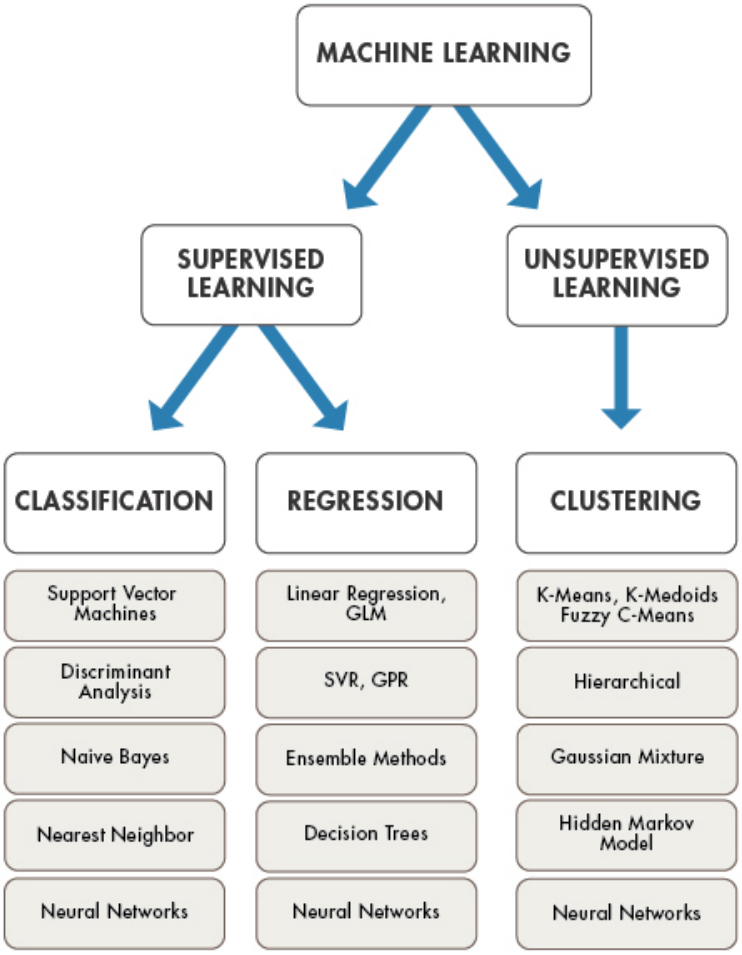


Figura 2.2: Algoritmi di Machine Learning più diffusi

una struttura dove le foglie rappresentano le etichette delle classi, mentre i rami rappresentano le congiunzioni di features che portano a queste etichette.

K-Nearest Neighbours

L'algoritmo K-Nearest Neighbours (K-NN) assume la classe di un determinato dato in ingresso come quella dei k elementi più "vicini" a lui nello spazio delle caratteristiche. L'algoritmo prevede come primo passo di calcolare la distanza tra le caratteristiche del dato da classificare e quelle dei dati usati per il training. Successivamente i dati di training e le rispettive distanze vengono disposte in ordine decrescente così da poter isolare i K dati più vicini. In questa maniera al nuovo dato verrà assegnata l'etichetta corrispondente alla classe con più rappresentanti all'interno dei K elementi prima selezionati. Se utilizzato per la regressione, allora la nuova etichetta sarà la media delle etichette associate ai K elementi più vicini. Il valore di K va scelto opportunamente. Per valori di K piccoli si avranno tempi di esecuzione minori ma poca precisione, che al contrario aumenta con il crescere di K. Come si vede anche in Figura 2.3 per K=3 si l'elemento in rosso verrà assegnato alla classe B, mentre aumentando K aumentera la precisione e l'elemento sarà assegnato alla classe A. Da un certo punto in poi, però, col crescere di K aumenteranno anche gli errori, segno che è stato superato il valore ottimo.

L'algoritmo presenta come principali vantaggi la semplicità e la versatilità, Inoltre non è necessaria la generazione di alcun modello. Lo svantaggio principale sta nel fatto che all'aumentare del numero di dati l'algoritmo diventi sempre più lento, rendendolo poco efficace per grandi dataset o per dati dotati di molte features.

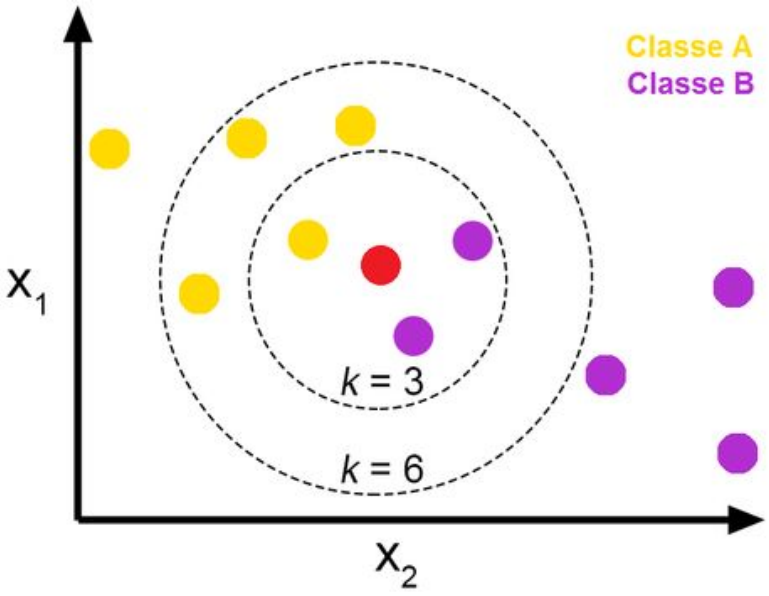


Figura 2.3: Esempio di classificazione mediante K-NN

Support Vector Machines

Le macchine a vettori di supporto (SVM) sono modelli di apprendimento supervisionato utilizzati sia per la regressione che per la classificazione. Dato un dataset di partenza, l'SVM ha l'obbiettivo di trovare un iperpiano in uno spazio N-dimensionale (dove N è il numero delle features dei nostri dati) che separi le classi di dati nella maniera più netta possibile, ovvero in maniera che si abbia la maggior distanza minima tra gli elementi delle diverse classi e il margine di separazione, come mostrato in figura 2.4.

Spesso, come nell'esempio in figura 2.5, non è possibile trovare un iperpiano che separi gli elementi delle diverse classi in uno spazio N-dimensionale. Diventa utile allora applicare il metodo kernel, una trasformazione non lineare che esegue la mappatura del nostro dataset in uno spazio a dimensione maggiore di N, dove sia possibile individuare un iperpiano che separi le nostre classi. Nella figura 2.6, gli elementi nell'esempio in figura 2.5 sono stati rimappati dallo spazio $\mathbb{R}^2$ a $\mathbb{R}^3$ attraverso la trasformazione non lineare $z = x^2 + y^2$. Come si può vedere dalla figura 2.7 è ora possibile trovare una retta (iperpiano in $\mathbb{R}^2$) che separi le due classi.

Nel corso della sperimentazione sono stati utilizzati kernel di tipo lineare, gaussiano e polinomiale.

Linear Discriminant Analysis

La Linear Discrimant Analysis è un metodo utilizzato nell'ambito del machine learning, o più in generale della data science, per determinare una combinazione lineare delle caratteristiche di un determinato dataset che possa caratterizzare o separare

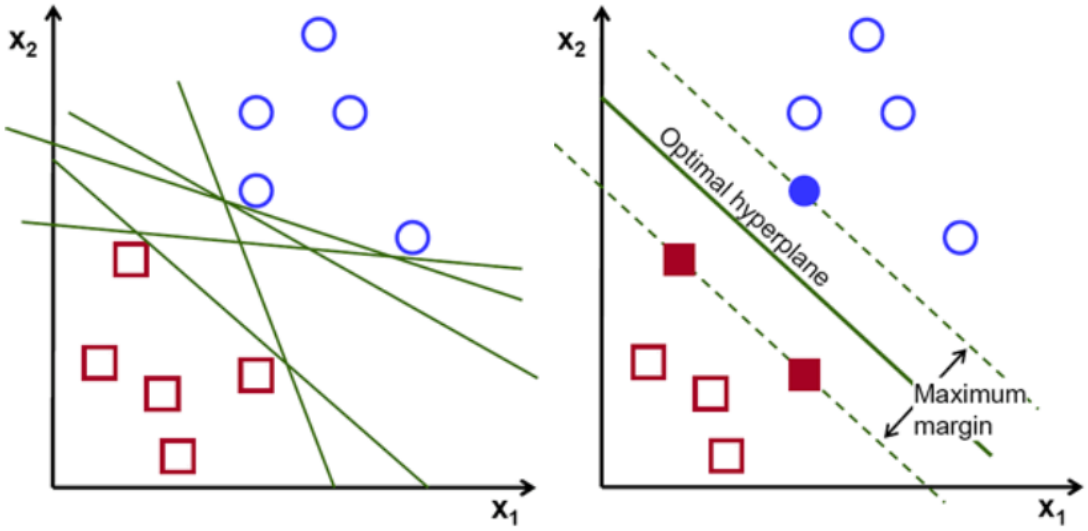


Figura 2.4: Esempio di classificazione mediante SVM

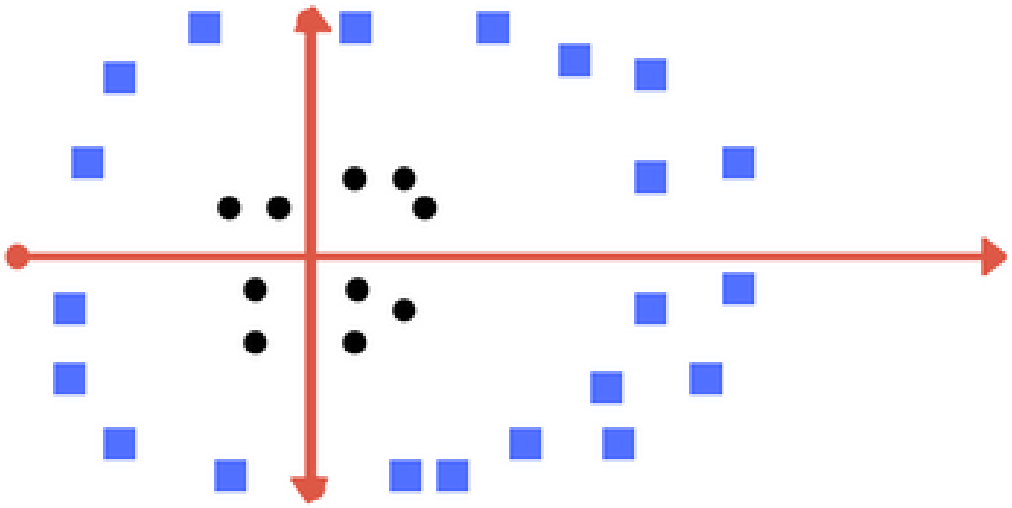


Figura 2.5: Dati non separabili in $\mathbb{R}^2$

due o più classi. In questo senso viene operata una riduzione di dimensioni del nostro dataset, passando da N componenti che caratterizzano ogni record ad una sola combinazione lineare di queste.

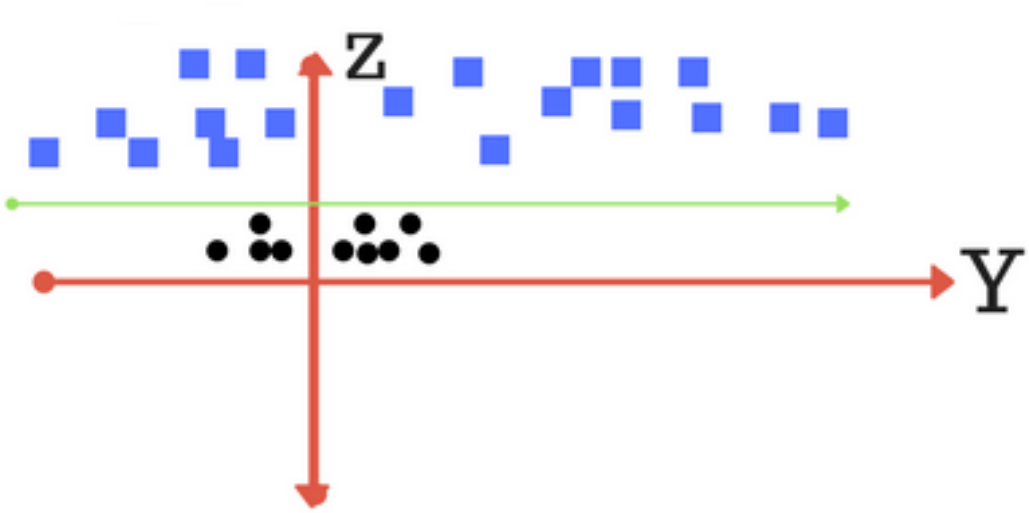


Figura 2.6: Grafico degli assi Z ed Y dopo l'applicazione del kernel non lineare

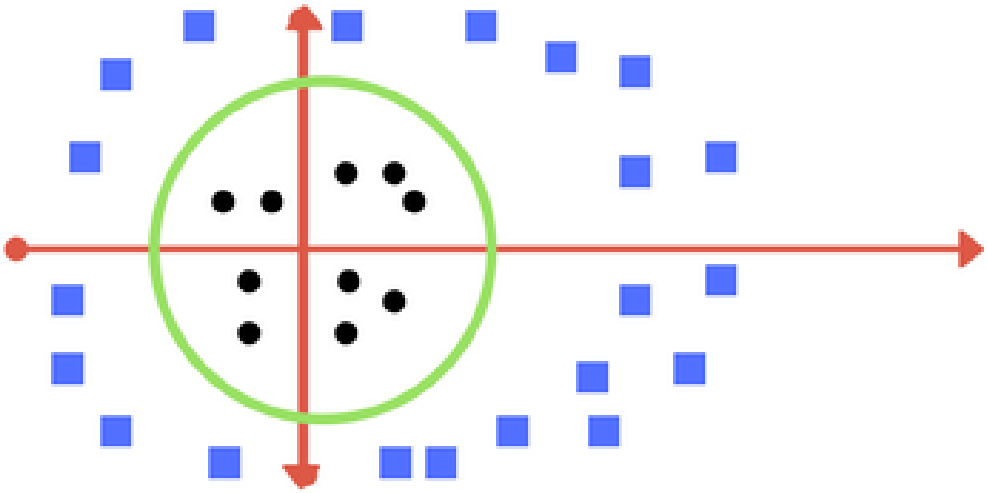


Figura 2.7: Trasformata inversa per tornare al piano xy

Capitolo 3

Dataset

3.1 Costruzione del Dataset

Il dataset utilizzato è stato realizzato dal tesista Leonardo Saraceni [3], acquisendo segnali fotopletismografici (PPG) e accelerometrici (ACC) alla frequenza di 400 Hz mediante il dispositivo della Maxim Integrated MAXREFDES100, posizionato al polso dei soggetti sottoposti alle misure. Il segnale fotopletismografico misura l'irrorazione sanguigna dei tessuti ed è ottenuto mediante lo studio della propagazione di un segnale luminoso attraverso le zone d'interesse. Il segnale accelerometrico misura invece il valore dell'accelerazione istantanea lungo i tre assi cartesiani. I dati sono stati prelevati sottoponendo 7 soggetti a 3 diversi test fisici. Il primo è stato lo step, il secondo lo squat ed infine sono stati acquisiti il sengale PPG e ACC a riposo per 5 minuti. I soggetti sono stati scelti per essere diversificati in età, peso e genere come riportato nella tabella 3.1. .

Altezza (m)	Peso (Kg)	Età (Anni)	Sesso
1,80	80	44	M
1,78	72	22	M
1,73	70	22	M
1,65	55	20	F
1,78	83	20	F
1,75	75	53	M
1,70	60	52	F
1,57	66	41	F

Tabella 3.1: Caratteristiche dei soggetti sottoposti ai test

Per tutte le prove, al fine di aumentare la quantità dei dati acquisiti, ogni attività è stata ripetuta e monitorata per 5 volte. Il dataset è quindi formato da 210 segnali, 105 di tipo fotopletismografico e 105 di tipo accelerometrico. I primi sono costituiti da un'unica componente, mentre i secondi ne presentane tre, una per asse cartesiano. L'acquisizione dei dati, inoltre, è avvenuta senza particolare attenzione alle tempistiche, per cui i diversi segnali presentano lunghezze differenti, che possono andare da 30 secondi a diversi minuti. A causa della natura ripetitiva dei due esercizi fisici proposti, i dati mostrano un'apparente periodicità. Nelle attività di squat e step, infatti, il segnale PPG presenta dei picchi circa ogni 200 campioni

mentre il segnale accelerometrico circa ogni 1200-1600 campioni, variabili a seconda dell'esercizio svolto.

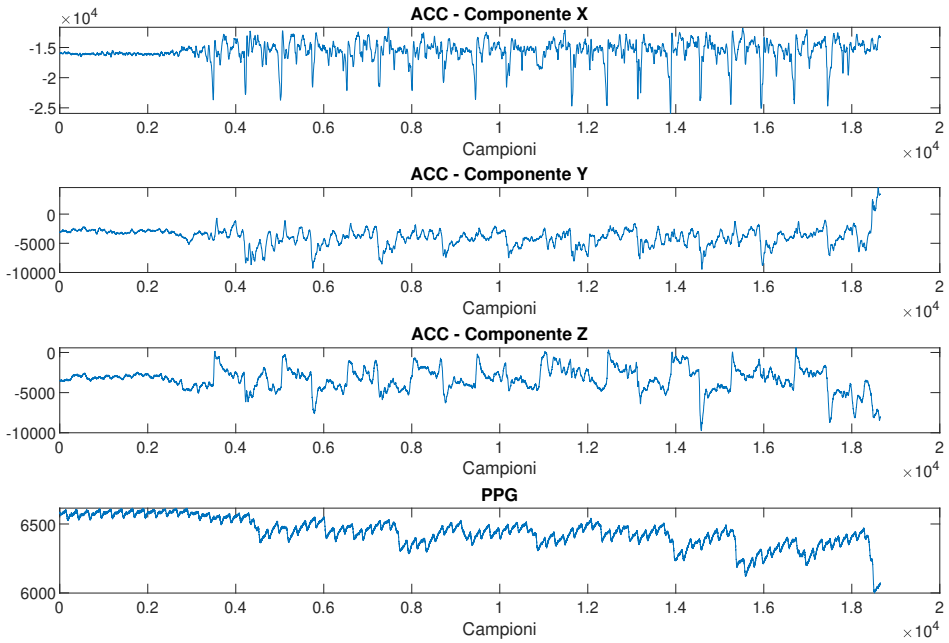


Figura 3.1: Segnali Accelerometrici e Fotopletismiografici relativi all'attività "Step" del soggetto 5

3.2 Elaborazione del Dataset

Il dataset è stato assegnato sottoforma di 210 diverse matrici. Le 105 relative ai segnali PPG erano costituite da 2 colonne, una contenente gli istanti temporali una i valori del segnale. Le restanti 105 relative al segnale ACC presentavano, oltre all'asse temporale, altre tre colonne, una per ogni direzione lungo la quale è stata misurata l'accelerazione. Per poter costruire la data matrix da utilizzare nella classificazione è stato necessario eliminare l'asse dei tempi di ogni matrice del dataset originale, ruotare la colonna dei dati ed applicare diverse finestre ai segnali per renderli di lunghezza identica. Le tre componenti del segnale accelerometrico sono state successivamente concatenate per creare un'unica riga nella data matrix. Nel procedimento sono anche state apposte in coda ad ogni record le etichette relative sia all'attività svolta sia al soggetto monitorato.

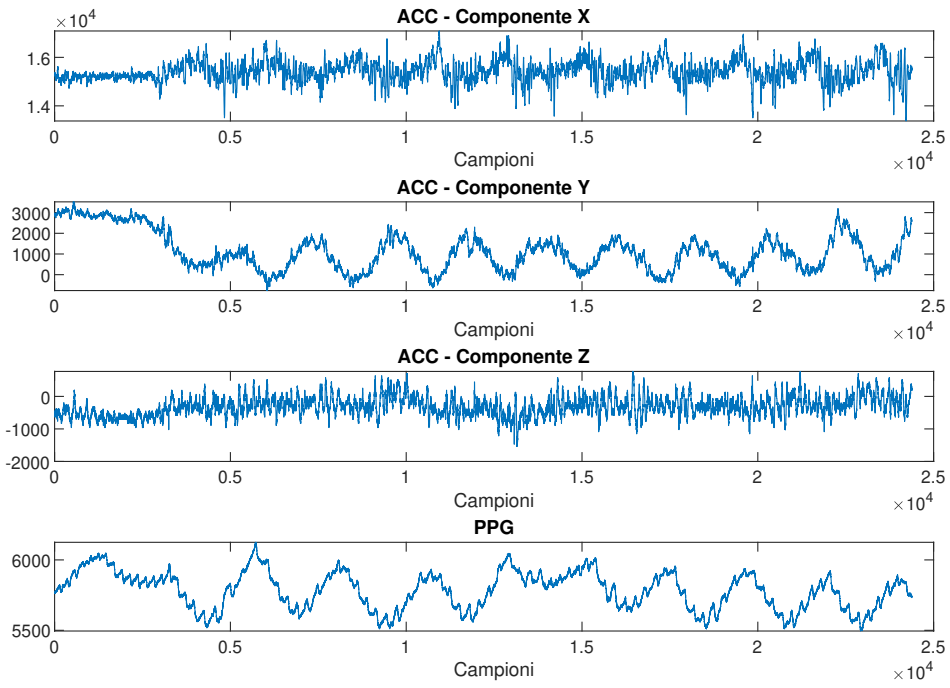


Figura 3.2: Segnali Accelerometrici e Fotopletismiografici relativi all'attività "Squat" del soggetto 5

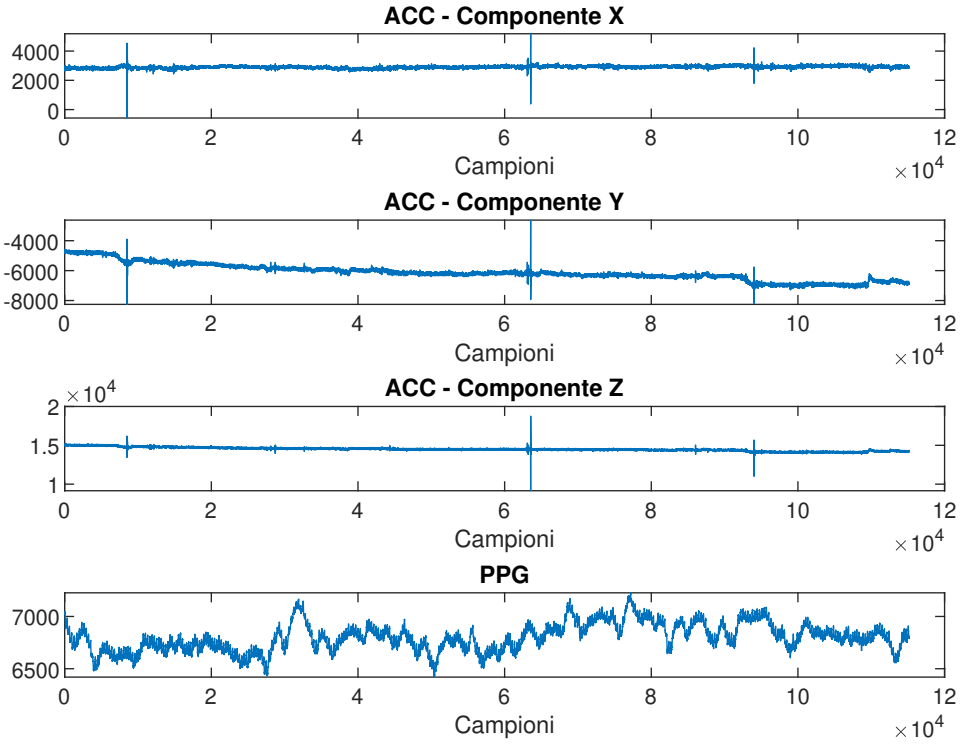


Figura 3.3: Segnali Accelerometrici e Fotopletismiografici del soggetto 5 a riposo

Capitolo 4

Sperimentazione

4.1 Validazione

Nella costruzione dei diversi modelli è stato necessario accertarsi che non si verificasse overfitting. L'overfitting è un problema che si verifica quando il modello generato è adattato troppo al dataset di training, modellandone anche l'eventuale rumore. Questo fa sì che si riesca bene a classificare i dati già contenuti nel dataset utilizzato per il training, ma si abbiano performance molto peggiori nella classificazione di nuovi dati. La presenza di overfitting può essere verificata attraverso il processo di validazione. Ne esistono di diversi tipi di e la maggior parte sfrutta l'idea di rimuovere una porzione del dataset di training e di utilizzarla per misurare le prestazioni del modello su dati a lui sconosciuti, così da poterne misurare la precisione. Esistono due principali metodi di validazione, La K-folds cross validation e l'holdout validation.

4.1.1 K-folds cross validation

La K-folds cross validation è un metodo che nasce dall'idea di di divedere il dataset in K parti dette folds. A turno, una delle sezione viene utilizzate per validare il dataset e tutte le altre, invece, sono utilizzate per allenare il modello. In questa maniera ogni porzione del dataset originale viene utilizzata una volta per la validazione e K-1 volte per il training. Ad ogni iterazione il modello generato viene utilizzato per classificare i dati della fold scelta per la validazione e l'accuracy viene determinata confrontando le classi predette con quelle reali. Alla fine del processo la precisione del modello è data dalla media delle diverse accuracy ottenute ad ogni iterazione. Il valore di k va stabilito a priori, tenendo conto che più questo è grande più si ha protezione dall'overfitting, a scapito dei tempi di calcolo che crescono tante più fold andiamo a creare. Solitamente viene utilizzato 10 come valore di K, in quanto risulta un buon compromesso tra la protezione da overfitting e l'onere computazionale.

4.1.2 Holdout Validation

L'holdout validation prevede che una porzione del dataset di partenza (in genere compresa tra il 20% e il 25% di tutti i dati) non venga utilizzata per il training, ma venga bensì riservata per la validazione del modello generato. In genere prima di operare la suddivisione del dataset è opportuno mescolare i dati, così che entrambe

le porzioni contengano una quantità sufficiente di elementi per ogni classe. Una volta generato il modello, quest'ultimo viene utilizzato per classificare i dati riservati per la validazione e confrontando le classi predette con le classi reali è possibile ottenere una misura dell'accuratezza del modello generato.

4.1.3 Validazione nella sperimentazione

Nella sperimentazione svolta nell'ambito di questa tesi si è deciso di utilizzare una particolare forma di holdout validation. Piuttosto che riservare il 20% del dataset di partenza per la validazione, si è scelto di utilizzare i soggetti 6 e 7. Questo ci consente di avere una migliore stima delle perfomance che avranno i modelli generati in situazioni reali, dove i dati da classificare proverranno da individui diversi da quelli utilizzati in fase di training.

4.2 Metodo di Test

Nel corso della sperimentazione tutti i modelli sono stati generati utilizzando le funzioni già contenute in Matlab che implementano gli algoritimi di Decision Tree, KNN, Linear SVM, Linear Discriminant e PCA, utilizzando opportuni parametri in ingresso. In alcuni casi non sono stati generati alcuni dei modelli a causa dei risultati non particolarmente notevoli e dei lunghi tempi di calcolo.

4.3 1° Caso di Studio

Prima di partire con la sperimentazione vera e propria, è stato necessario costruire la data matrix. Per far questo è stano necessario ricavare dai dati di partenza dei record tutti della stessa lunghezza. Si è deciso, nel primo caso di studio, di troncare tutti i segnali alla lunghezza del più corto, corrispondente a 12974 campioni e di concatenare le tre componenti del segnale accelerometrico.

Tabella 4.1: Caso di Studio N°1

Tipo di Segnale	Tree	K-NN	Linear Discriminant	Linear SVM
PPG	43,33%	36,67%	40%	53,33%
ACC	66,67%	60%	*	50,00%

* Non calcolato a causa delle dimensioni troppo elevate della matrice

I risultati mostrati nella tabella 4.1 sono mediocri, ma i modelli derivati dal segnale accelerometrico performano meglio, a causa della maggior lunghezza del singolo record, dovuta alla concatenazione delle tre componenti. Le perfomance pessimo possono essere spiegate guardando le dimensioni delle due data matrix, in quanto presentano molte colonne e poche righe. Queste dimensioni non sono ideali per eseguire la classificazione ed è stato quindi necessario aumentare il numero totale dei record.

4.4 2° Caso di Studio

Per accrescere il numero di record totali, le righe della data matrix del caso di studio precedente sono stati divisi in finestre di lunghezza arbitrarie, scelte tra i divisori interi di 12974, che fossero compatibili con la periodicità apparente del segnale.

Tabella 4.2: Caso di Studio N°2

Tipo Di Segnale	L Finestra	Tree	K-NN	Linear Discriminant	Linear SVM
PPG	998	36,66%	35,64%	29,49%	37,43%
PPG	2162	32,22%	32,78%	39,44%	40,00%
ACC	998	55,21%	54,70%	36,15%	32,82%
ACC	2162	45,00%	55,92%	35,00%	37,78%

I risultati sono peggiorati, probabilmente a causa della scarsa compatibilità delle lunghezze di finestra utilizzabili con la periodicità apparente del segnale, stimata a circa 1200-1600 campioni.

4.5 3° Caso di Studio

Nei due casi di studio precedenti,a causa del tipo di troncamento operato, gran parte dei dati acquisiti non è stata utilizzata. Si è pensato quindi di ricavare i record di una nuova data matrix suddividendo i dati originali in porzioni di lunghezza uguale e arbitraria, scartando eventuali rimanenze causate dal fatto che la lunghezza della finestra non divida interamente la lunghezza del segnale.

Tabella 4.3: Caso di Studio N°3 - Segnale PPG

L Finestra	Tree	K-NN	Linear Discriminant	Linear SVM	Gaussian SVM
700	59,42%	66,07%	72,64%	73,13%	69,60%
800	64,70%	67,07%	70,85%	73,17%	70,20%
900	65,21%	66,23%	71,33%	73,23%	71,22%
1000	60,87%	65,72%	69,75%	73,16%	71,46%
1100	66,67%	65,83%	69,79%	73,26%	73,06%
1200	58,73%	65,80%	68,62%	73,26%	73,10%
1300	63,21%	65,02%	65,18%	73,33%	72,59%
1400	64,35%	68,00%	65,33%	73,42%	73,07%
1500	62,55%	65,97%	64,07%	73,48%	73,10%
1600	59,95%	66,36%	61,17%	73,48%	73,27%
1700	58,25%	66,45%	63,86%	73,57%	73,35%
1800	57,27%	66,09%	56,70%	73,54%	73,42%
1900	65,86%	66,71%	54,00%	73,73%	73,73%
2000	58,78%	67,05%	49,74%	73,54%	73,41%

* SVM Polinomiale non utilizzata a causa dei tempi elevati di elaborazione

Come mostrano le tabelle 3.3 e 3.4, i risultati sono nettamente migliorati. Osservando la tabella 3.4 relativa alla classificazione del segnale accelerometrico, si noti come i modelli basati su Decision Tree, KNN, Linear SVM si comportino meglio,

Tabella 4.4: Caso di Studio N°3 - Segnale ACC

L Finestra	Tree	K-NN	Linear Disc.	Lin. SVM	Gauss. SVM	Poly SVM
700	73,96%	83,21%	67,66%	83,48%	73,13%	78,94%
800	85,53%	83,11%	65,10%	83,66%	73,17%	76,15%
900	80,24%	84,05%	59,48%	82,69%	73,21%	76,99%
1000	83,80%	81,03%	58,34%	82,04%	73,15%	75,55%
1100	71,11%	84,10%	51,25%	82,57%	73,25%	75,38%
1200	71,05%	81,92%	39,66%	82,75%	73,33%	73,91%
1300	71,77%	83,62%	42,88%	82,55%	74,42%	75,64%
1400	84,44%	82,40%	50,93%	82,49%	73,48%	73,95%
1500	83,36%	83,27%	57,70%	82,32%	73,47%	76,32%
1600	84,05%	81,50%	64,13%	83,03%	73,57%	76,05%
1700	83,06%	81,88%	69,26%	82,52%	73,54%	76,40%
1800	70,10%	81,44%	70,22%	82,47%	73,73%	76,51%
1900	83,17%	81,96%	71,19%	82,81%	73,54%	74,93%
2000	78,63%	81,42%	74,30%	82,32%	73,26%	75,56%

mentre quelli realizzati mediante Linear Discriminant performino leggermente peggio. I modelli basati su SVM gaussiana e polinomiale hanno buoni risultati, ma peggiori rispetto a quelli dati dall'SVM lineare, per cui si è preferito utilizzare solo questa per i casi di studio successivi.

4.6 4° Caso di Studio

Per migliorare ancora le prestazioni dei modelli si è deciso di sfruttare contemporaneamente il segnale accelerometrico e il fotopletismogramma, creando una nuova data matrix di partenza semplicemente affiancando la due matrici del caso di studio precedente. Non sono stati generati i modelli mediante Linear Discriminant, in quanto poco performanti e molto onerosi dal punto di vista del tempo di computazione.

Come si vede nella tabella 3.6, l'accuracy dei modelli basati su Decision Tree è migliorata leggermente, mentre è peggiorata quella dei modelli basati su KNN e Linear SVM. Le finestre migliori sono concentrate tra i 1400 e i 1700 campioni, in accordo con la periodicità apparente del segnale accelerometrico.

4.7 5° Caso di Studio

Poichè i due tipi di segnale sono su scale leggermente diverse, si è pensato di normalizzare il segnale PPG e le tre componenti del segnale accelerometrico prima di fondere le due data matrix, utilizzando la normalizzazione $\hat{X}_{\mathrm{i}} = \frac{X_{\mathrm{i}} - \mu}{\sigma}$, dove μ è il valor medio del segnale o della singola componente e σ la sua deviazione standard. I nuovi segnali ottenuti saranno a valor medio nullo e con deviazione standard pari ad 1.

Come mostra la tabella 3.7, l'accuracy dei modelli peggiora soprattutto per il Decision Tree. Migliorano leggermente,invece, le performance dei modelli basate su

Tabella 4.5: Caso di Studio N°4 - Data Fusion

Lunghezza Finestra	Decision Tree	K-NN	Linear SVM
700	77,04%	78,37%	71,90%
800	90,41%	78,56%	71,71%
900	80,76%	79,51%	70,71%
1000	88,40%	79,33%	70,06%
1100	71,46%	78,26%	70,90%
1200	71,73%	76,52%	70,82%
1300	71,76%	77,44%	70,12%
1400	85,06%	77,95%	70,13%
1500	87,74%	78,04%	70,15%
1600	87,70%	77,85%	70,53%
1700	87,05%	78,75%	69,90%
1800	70,10%	77,89%	69,87%
1900	83,65%	77,24%	69,25%
2000	78,88%	79,00%	69,97%

Tabella 4.6: Caso di Studio N°5 Normalizzazione

Lunghezza Finestra	Decision Tree	K-NN	Linear SVM
700	75,02%	75,86%	73,13%
800	74,43%	75,49%	73,17%
900	71,23%	77,30%	73,21%
1000	76,18%	77,63%	73,22%
1100	74,44%	77,36%	74,10%
1200	73,25%	78,72%	73,25%
1300	76,79%	78,02%	73,58%
1400	76,09%	77,42%	74,04%
1500	73,57%	76,05%	74,14%
1600	76,82%	76,63%	74,19%
1700	77,78%	77,24%	75,40%
1800	73,08%	76,52%	74,68%
1900	78,09%	76,27%	75,42%
2000	74,42%	75,44%	74,80%

Linear SVM. Nel complesso la normalizzazione aggiuntiva non provoca miglioramenti sufficienti a giustificarne l'applicazione, per cui non è stata riutilizzata nei casi di studio successivi.

4.8 6° Caso di Studio

Per far sì che i diversi segmenti di segnale si adattassero meglio alla periodicità di lunghezza variabile dei dati si è deciso di applicare le finestre con una sovrapposizione parziale. Poichè con questa operazione viene aumentato il numero dei record totali, ci si aspetta che anche l'accuracy media delle diverse finestre sia più alta.

Tabella 4.7: Caso di Studio N°6 - Finestratura con Overlap

Lunghezza Finestra	Overlap	Decision Tree	K-NN	Linear SVM
800	100	75,24%	78,87%	71,26%
800	200	84,47%	78,56%	71,75%
900	100	88,90%	78,30%	71,19%
900	200	77,02%	78,79%	71,18%
1000	100	84,81%	79,40%	70,70%
1000	200	89,41%	78,43%	79,86%
1000	300	77,27%	78,42%	70,75%
1100	100	88,57%	79,29%	70,45%
1100	200	84,86%	79,58%	70,93%
1200	200	88,29%	78,92%	37,21%
1200	300	71,68%	79,22%	71,00%
1200	400	88,83%	78,53%	70,77%
1300	200	70,76%	78,15%	71,04%
1300	300	88,27%	78,69%	70,07%
1300	400	85,36%	78,79%	70,69%
1300	500	88,84%	79,02%	70,51%
1400	200	76,12%	77,42%	70,25%
1400	300	70,58%	77,56%	70,71%
1400	400	88,78%	78,28%	69,68%
1400	600	89,54%	78,73%	70,37%
1400	800	79,70%	78,32%	70,11%
1500	100	85,11%	72,63%	70,14%
1500	200	72,11%	76,89%	69,72%
1500	300	75,38%	77,21%	70,26%
1500	400	75,45%	77,06%	70,20%
1600	100	85,63%	77,93%	69,93%
1600	200	86,16%	77,32%	70,08%
1600	300	72,81%	76,86%	70,00%
1700	200	82,92%	77,77%	70,32%
1700	300	85,79%	77,30%	70,42%
1800	100	87,13%	78,92%	70,27%
1800	200	82,36%	77,47%	70,74%
1800	300	82,97%	77,13%	70,62%
1900	200	85,06%	78,79%	69,91%
1900	300	83,38%	77,68%	70,74%
2000	200	70,38%	77,84%	70,26%
2000	300	85,26%	78,87%	70,21%

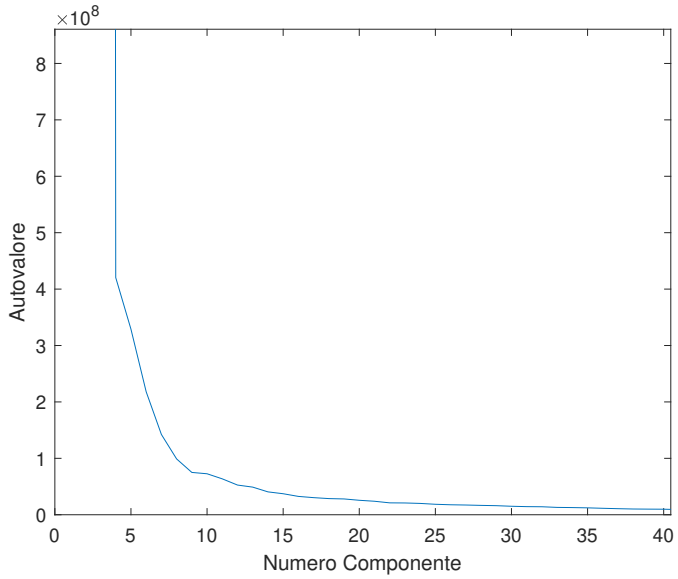
Come si evince dalla tabella 3.8, le performance non migliorano particolarmente, anche se per alcune specifiche lunghezze di finestra (1200,1300,1800) l'overlap ha migliorato l'accuracy del modello fino al 18% rispetto a quella del caso di studio precedente.

4.9 7° Caso di Studio

Poichè ogni record, anche dopo aver applicato la segmentazione dei dati, possiede un numero di features nell'ordine delle migliaia, si è deciso di applicare la PCA, al fine di ridurre le dimensioni della nostra data matrix. La PCA (Principal Compenents Analysis) è una tecnica che consente di estrarre da un dataset con molte componenti correlate tra loro altrettante componenti incorrelate, ottenute calcolando gli autovettori della matrice di covarianza della nostra data matrix. Utilizzando solo le componenti con il maggior contenuto informativo per il training è possibile ridurre la dimensione del dataset senza intaccare troppo l'accuratezza del modello.

Rappresentando gli autovalori della matrice di covarianza si evince che la maggior parte del contenuto informativo sia contenuto nelle prime 30 componenti.

Figura 4.1: Autovalori della matrice di covarianza per dati con finestra di 1400 Campioni e Overlap di 600 Campioni



Le nuove data matrix sono state quindi generate utilizzando dalle 5 alle 40 componenti calcolate a partire dalle matrici più performanti dei casi di studio precedenti. Non sono stati generati i modelli mediante Linear SVM in quanto necessitano di lunghi tempi di calcolo e offrono risultati peggiori rispetto agli altri modelli.

Tabella 4.8: Feature Reduction - Finestra da 800 Campioni senza Overlap

N° Componenti	Decision Tree	K-NN
5	73,27%	76%
10	76,40%	83,00%
15	75,19%	84,52%
20	75,09%	93,60%
25	75,14%	93,39%
30	74,98%	92,94%
35	75,14%	91,68%
40	75,14%	90,37%

Tabella 4.9: Feature Reduction - Finestra da 1200 Campioni con 400 campioni di Overlap

N° Componenti	Decision Tree	K-NN
5	70,38%	73,67%
10	71,39%	92,40%
15	71,77%	92,72%
20	71,58%	91,96%
25	71,58%	93,42%
30	71,52%	93,29%
35	71,71%	92,60%
40	71,58%	92,47%

Tabella 4.10: Feature Reduction - Finestra da 1400 Campioni con 600 campioni di Overlap

N° Componenti	Decision Tree	K-NN
5	74,30%	74,25%
10	74,50%	83,38%
15	75,53%	94,70%
20	75,42%	95,46%
25	75,42%	95,00%
30	75,88%	94,85%
35	75,88%	94,04%
40	75,88%	94,13%

Come si può vedere dai risultati nelle tabelle 4.9, 4.10 e 4.11, si ha un notevole miglioramento dell'accuracy utilizzando K-NN. In particolare le migliori prestazioni si hanno prendendo tra le 20 e le 25 componenti. Questo è presumibilmente dovuto all'azione di filtraggio operato dalla PCA. Le due attività di movimento, infatti, avvegono lungo una principale direzione, mentre il resto dell'informazione risulta non utile se non dannoso ai fini della classificazione. La PCA riesce ad estrarre in maniera più efficace le componenti con il maggior contenuto informativo e ad eliminare tutto il restante contributo che risulta rumoroso, facendo così crescere di molto l'accuracy.

4.10 8° Caso di Studio

Questo caso di studio è stato riportato per evidenziare come una delle componenti possa essere più informativa delle altre e come l'utilizzo di tutta l'informazione in maniera non discriminata non porti sempre ad un miglioramento dell'accuratezza. Nel corso degli altri casi di studio sono stati infatti accidentalmente generati modelli partendo dal segnale PPG e dalla solo componente X del segnale accelerometrico. I risultati sono sorprendentemente buoni, come mostrato nella tabella 4.11.

Confrontando i risultati con quelli ottenuti utilizzando tutte e tre le componenti del segnale accelerometrico possiamo notare come ci sia un miglioramento di circa il 10% nell'accuracy dei modelli basati su KNN, mentre quelli basati su Decision Tree

Tabella 4.11: Caso di Studio N°8 - PPG e sola componente X

Lunghezza Finestra	Overlap	Decision Tree	K-NN
900	100	91,17%	88,45%
900	200	80,26%	88,80%
1000	0	82,99%	88,79%
1000	100	91,41%	88,28%
1000	200	91,18%	88,35%
1000	300	82,98%	88,79%
1100	0	82,99%	87,85%
1100	100	92,30%	87,50%
1100	200	91,69%	88,62%
1100	300	91,23%	88,19%
1100	400	83,04%	88,64%
1200	0	83,73%	87,99%
1200	100	82,96%	87,76%
1200	200	92,27%	87,59%
1200	300	90,52%	88,58%
1200	400	90,87%	87,77%
1200	500	82,44%	88,71%
1200	600	90,23%	88,16%
1200	900	91,40%	88,55%
1300	0	79,91%	87,65%
1300	100	83,78%	87,20%
1300	200	79,34%	87,79%
1300	300	91,50%	87,63%
1300	400	91,88%	88,79%
1300	500	92,00%	87,22%
1400	100	80,64%	87,56%
1400	200	83,60%	87,80%
1400	300	78,69%	88,05%
1400	400	91,97%	87,77%
1400	500	91,80%	88,64%
1400	600	92,25%	87,45%

abbbiano performance leggermente migliori. Questo dimostra in maniera ancora più evidente come sia necessario operare un filtraggio sul segnale accelerometrico al fine di rimuovere il contenuto rumoroso e mantenere soltanto le componenti più significative.

4.11 9° Caso di Studio

In questo caso di studio sono stati utilizzati due ulteriori algoritmi: il Particle-Bernstein Polynomials e il Monte Carlo.

Il Particle-Bernstein Polynomials (PBP) è stato sviluppato all'interno del Dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche. È descritto nell' articolo *"Machine learning regression based on particle bernstein*

poly-nomials for nonlinear system identification" [1] ed già stato utilizzato con successo nella classificazione del morbo di Alzheimer sfruttando i segnali ottenuti da risonanze magnetiche [2]. L'algoritmo sfrutta i polinomi di Bernstein di grado m nell'intervallo [0, 1], dati dalla formula

$$b_k^m(x) = \binom{m}{k} x^k (1-x)^{m-k} \qquad k = 0, 1, ..., m \tag{4.1}$$

dove x è compreso tra 0 e 1 e il coefficiente binomiale $\binom{m}{k}$ è dato da $\frac{m!}{k!(m-k)!}$. Si può dimostrare che, per ogni funzione continua $f(x)$ in [0, 1] la sequenza $B_{\mathrm{m}}(x)$

$$B_{\mathrm{m}}(x) = \sum_{k=0}^{m} f(k/m) b_k^m(x) \tag{4.2}$$

converge uniformemente ad f per $m \to \infty$. Questo implica che $B_{\mathrm{m}}(x)$ è in grado di approssimare la funzione $f(x)$ con precisione crescente al crescere di m, supponendo che si conosca la funzione f nei punti $f(k/m)$, per $k = 0, 1, ..., m$. Sfruttando questa proprietà è possibile ottenere un algoritmo di regressione utilizzabile, attraverso opportuni accorgimenti, anche per la classificazione.

L'algoritmo Monte Carlo [6] è stato sviluppato dal professor Claudio Turchetti e dalla dottoressa Laura Falaschetti del Dipartimento di Ingegneria dell'Informazione dell'Università Politecnica delle Marche. L'algoritmo di classificazione si basa appunto sul metodo Monte Carlo, che nasce dall'idea di ripetere un campionamento casuale al fine di ottenere risultati numerici. Viene utilizzato in molti contesti per risolvere con la casualità problemi di natura deterministica come integrali numerici o ottimizzazioni.

Per verificare le prestazioni dei due algoritmi è stato deciso di applicare entrambi alla data matrix costruita applicando finestre arbitrarie ai due tipi di segnali senza overlap (come nel caso di studio $N°4$) e di confrontare i risultati con quelli ottenuti utilizzando il K-NN. Per migliorare le prestazioni, inoltre, è stata applicata la PCA per isolare le 20 componenti più significative. Al fine di evitare errori di overflow, prima di utilizzare i due nuovi algoritmi è stato necessario normalizzare le componenti estratte nel range [0, 1].

Come si vede nella tabella 4.12 i due nuovi algoritmi hanno performance media migliore dell'1-1,5% rispetto a quella del K-NN. Inoltre, osservando la riga evidenziata, si può notare come PBP e Monte Carlo riescano a raggiungere precisioni più elevate, superando persino il 97%.

Tabella 4.12: Caso di Studio N°9 - PBP e Montecarlo Vector

Lunghezza finestra	PBP	Monte Carlo	K-NN
700	92,11%	91,54%	87,53%
800	89,20%	89,06%	93,60%
900	95,74%	94,72%	93,53%
1000	92,25%	92,12%	93,44%
1100	95,90%	94,24%	93,89%
1200	95,21%	95,29%	93,69%
1300	96,46%	96,21%	93,25%
1400	94,04%	94,31%	93,07%
1500	96,10%	94,58%	94,68%
1600	95,73%	96,04%	92,89%
1700	94,50%	93,31%	93,85%
1800	96,68%	96,91%	93,70%
1900	97,21%	96,49%	93,95%
2000	96,56%	95,80%	93,13%
Average	94,84%	94,33%	93,16%

Capitolo 5

Conclusioni

Il lavoro svolto dimostra come sia possibile ottenere ottimi modelli di classificazione partendo dal dataset descritto nell'articolo *"Dataset from PPG wireless sensor for activity monitoring"*[3], utilizzando diverse tecniche di machine learning. In particolare i risultati migliori sono stati ottenuti utilizzando l'algoritmo K-NN e i due algoritmi sperimentali Monte Carlo e PBP, avendo però applicato prima la PCA, al fine di ridurre la dimensionalità della nostra data matrix e di rimuovere tutto il contenuto rumoroso all'interno del segnale accelerometrico. La PCA ha anche provveduto a ridurre i tempi necessari ad allenare il modello e a realizzare la classificazione. Questo rende più plausibile anche l'implementazione del nostro classificatore su dispositivi mobili o indossabili, rendendo un'ipotesi non inverosimile la classificazione operata in tempo reale. Spesso infatti, l'acquisizione e l'elaborazione dei dati avvengono su dispositivi separati. La prima viene realizzata dai sensori che operano a livello locale, per esempio in un'automobile o indossati da un atleta. L'elaborazione, invece, avviene a livello superiore nella maggior parte dei casi, e solitamente viene eseguita su un server, dove convergono anche i dati provenienti da molti altri sensori. Questo approccio presenta diversi problemi, primo tra tutti la latenza nella comunicazione tra sensore e server, che può essere tollerata in alcune applicazioni ma può essere decisiva in altre. Un esempio è quello della guida automatica delle autovetture, dove il riconoscimento di ostacoli deve essere operato in frazioni di secondo per evitare incidenti. Un altro problema sta nella capacità computazionali del server, che non necessariamente saranno in grado di stare al passo con la quantità sempre crescente di dati da processare. Da qui nasce il vantaggio di dividere l'elaborazione portandola al livello più basso dei sensori che acquisiscono i dati. Questo rende fondamentale avere classificatori che possano essere eseguiti sfruttando le limitate capacità di memoria di cui si dispone a livello locale, premiando il risultato finale, in cui la riduzione di complessità implica anche un miglioramento nelle prestazioni.

Per ottenere ulteriori progressi si è pensato a diverse possibili strade da percorrere. La prima è quella di ampliare il dataset di partenza. I dati relativi a 7 soggetti, infatti, costituiscono una buona base per la sperimentazione ma è altamente plausibile che si avrebbero migliori perfomance dei diversi modelli se fosse possibile allenarli sfruttando una quantità maggiore di dati provenienti da nuovi soggetti. Alcuni degli eventuali nuovi segnali integrati nel dataset, inoltre, potrebbero essere usati per la validazione, consentendo di ottenere una stima ancora migliore dell'accuracy dei modelli generati. Un'altra strada che rimane aperta è quella di utilizzare ulteriori tecniche di data fusion. La semplice concatenazione dei due tipi di

segnali utilizzata nell'ambito del lavoro svolto dà già degli ottimi risultati, ma non è da escludere che possano esistere metodi migliori per sfruttare contemporaneamente il segnale fotopletismografico e quello accelerometrico. Allo stesso modo sarà possibile, per chiunque voglia proseguire il lavoro svolto, sperimentare utilizzando algoritmi diversi da quelli utilizzati in questa tesi. Come ultimo sviluppo potrebbe essere interessante verificare la possibilità di implementare effettivamente il classificatore su un dispositivo in grado anche di acquisire il segnale accelerometrico e fotopletismografico, al fine di tentare di realizzare una classificazione in tempo reale dell'attività svolta dal soggetto esaminato.

Bibliografia

[1] G. Biagetti et al. «Machine learning regression based on particle bernstein polynomials for nonlinear system identification». In: *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. 2017, pp. 1–6. DOI: 10.1109/MLSP.2017.8168148.

[2] Giorgio Biagetti et al. «Classification of Alzheimer's Disease from Structural Magnetic Resonance Imaging using Particle-Bernstein Polynomials Algorithm». In: *Intelligent Decision Technologies 2019*. A cura di Ireneusz Czarnowski, Robert J. Howlett e Lakhmi C. Jain. Singapore: Springer Singapore, 2019, pp. 49–62.

[3] Giorgio Biagetti et al. «Dataset from PPG wireless sensor for activity monitoring». In: *Data in Brief* 29 (2020), p. 105044. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2019.105044. URL: http://www.sciencedirect.com/science/article/pii/S2352340919314003.

[4] Leonardo Saraceni. «Acquisizione del segnale fotopletismografico (PPG) nel corso di diverse attività fisiche tramite il dispositivo della Maxim Integrated MAXREFDES100 e classificazione delle attività svolte tramite algoritmi di machine learning.» B.S. Thesis. Università Politecnica delle Marche.

[5] *Towards Data Science - Descrizione algoritmi, esempi ed immagini*. https://towardsdatascience.com/.

[6] C. Turchetti e L. Falaschetti. «A GPU Parallel Algorithm for Non Parametric Tensor Learning». In: *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. 2018, pp. 286–290. DOI: 10.1109/ISSPIT.2018.8642737.

[7] *Wikipedia - Decision Tree Learning*. https://en.wikipedia.org/wiki/Decision_tree_learning.