

UNIVERSITÀ DEGLI STUDI DI UDINE

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Triennale in Tecnologie Web e Multimediali

Tesi di Laurea

INFERENZA DEI MEZZI DI TRASPORTO DEGLI UTENTI UTILIZZANDO DATI SENSORIALI MULTIPLI IN MODO NON PARAMETRICO

Relatore:

Dott. IVAN SCAGNETTO

Laureando:

MICHAEL SOPRANO

Correlatore:

Dott. MARCO PAVAN

ANNO ACCADEMICO 2013-2014

Ai miei genitori ed a mio fratello.

Ringraziamenti

Quella che state per leggere, probabilmente, è stata la parte più divertente e spensierata scritta nella tesi dove, almeno per una volta, non sono state necessarie ore passate a programmare o a leggere documenti. Il mio desiderio, infatti, è spendere qualche parola per ringraziare tutte quelle persone che hanno contribuito a rendere più divertenti questi anni passati tra università e casa dello studente supportando, inoltre, il raggiungimento di questo primo grande traguardo nel percorso di studi che ho deciso di intraprendere, poiché sono certo che la loro presenza è stata fondamentale per farmi diventare chi sono ora. Alcune di queste persone sono miei amici fin dai tempi delle scuole superiori, alcune sono compagni di corso, altre sono coinquilini in casa dello studente, o anche tutto ciò allo stesso tempo.

Un sentito grazie, dunque, a Filippo, Veronica, Andrea, Elisa, Jacopo, Brigita, Francesco, Giulio, Luca V., Luca F., Federico, Giuliano, Caterina, David, Daniele, Alice, Pescu, Eleonora e Johnny. Sono sicuro che ognuno degli interessati sarà in grado di riconoscersi in questo elenco, che non ha certo la pretesa di essere esaustivo. Intendo scusarmi in anticipo con chi mi sono dimenticato di inserire.

Aggiungo, inoltre, un ringraziamento ai dottori Ivan Scagnetto e Marco Pavan, rispettivamente relatore e correlatore di questa tesi, per avermi indirizzato e guidato lungo il percorso che ha portato alla realizzazione di tutto quello che costituisce il lavoro descritto. Infine, ringrazio anche la mia famiglia, semplicemente per avermi permesso di intraprendere tale percorso.

Michael

Indice

Ringraziamenti	v
Indice	vii
Elenco delle figure	ix
Elenco delle tabelle	xi
1 Introduzione	1
1.1 Ambito	1
1.2 Obiettivo	2
1.3 Stato dell'arte	3
1.3.1 Machine Learning	3
1.3.2 Data Mining	5
1.3.3 Statistica	6
1.3.4 Accelerometro	8
1.3.5 Giroscopio	9
1.3.6 Magnetometro	10
1.3.7 GPS	12
1.4 Tecnologie utilizzate	14
1.4.1 Java	14
1.4.2 Java FX	15
1.5 Altri lavori	16
1.6 Sinossi	18
2 Elaborazione dei dati grezzi	23
2.1 Dataset utilizzato	23
2.2 Preparazione dei dati	24
3 Classificazione	33
3.1 Concetti base	33
3.2 Estrazione delle feature	34

3.3	Weka	38
3.4	Classificatori basati sulle regole	41
3.4.1	OneR	42
3.5	Classificatori basati su alberi	43
3.5.1	J48	44
3.5.2	RandomForest	44
3.6	Classificatori Bayesiani	45
3.6.1	NaiveBayes	46
3.6.2	BayesNet	48
3.7	Reti neurali	50
3.8	Macchine a vettori di supporto	52
3.8.1	Maximum-Margin Hyperplane	53
3.9	Analisi risultati	54
4	Clustering	65
4.1	Concetti di base	65
4.2	Operazioni sui dati	67
4.3	Dirichlet Process Mixture Model	75
4.4	Hierarchical Dirichlet Process	79
4.5	Affinity Propagation	80
4.6	Analisi risultati	82
5	Conclusioni	93
5.1	Sviluppi futuri	94
5.2	Grafici e dati	96
5.2.1	Geolife	97
5.2.2	Twente	111
5.2.3	Lifetracker	129
	Bibliografia	141

Elenco delle figure

1.1	Penetrazione degli smartphone calcolata negli stati più tecnologicamente avanzati del pianeta – Febbraio 2013 (Fonte: Nielsen Global Smartphone Insights)	19
1.2	Percentuali di utilizzo delle tre tipologie di cellulare – Febbraio 2013 (Fonte: Nielsen Global Smartphone Insights)	20
1.3	Percentuali di utilizzo di smartphone e feature phone suddivise per sesso ed età – Febbraio 2013 (Fonte: Nielsen Global Smartphone Insights)	21
1.4	Passaggi fondamentali nella realizzazione del lavoro proposto. . .	22
2.1	Riga d'esempio tratta da un file del dataset utilizzato.	23
2.2	Prima finestra dell'interfaccia grafica del software desktop. . . .	27
2.3	Gerarchia delle classi che modellano i quattro sensori.	28
2.4	Schema dei metodi chiamati dalle istanze dei sensori per elaborare i dati grezzi.	28
2.5	Esempio di grafico relativo alla magnitudo dell'accelerometro. . .	29
2.6	Esempio di grafico relativo alla magnitudo del giroscopio.	29
2.7	Esempio di grafico relativo alla magnitudo del magnetometro. . .	30
2.8	Esempio di grafico relativo ai valori della velocità calcolati dal GPS.	30
2.9	Seconda finestra dell'interfaccia grafica del software desktop. . .	31
2.10	Schema del modulo di parsing dei dataset pubblici.	32
3.1	Schema delle chiamate effettuate per l'estrazione delle feature. . .	35
3.2	Schema delle chiamate effettuate per scrivere le feature estratte in un file ARFF.	39
3.3	Esempio di albero di decisione costruito da J48.	45
3.4	Esempio di dataset per Naive Bayes.	46
3.5	Esempio di DAG costruito da BayesNet.	49
3.6	Definizione di un Perceptron.	50
3.7	Struttura del MultilayerPerceptron.	51

3.8	Esempio di punti linearmente separabili.	53
3.9	Esempio di Maximum-Margin Hyperplane - (Fonte: [1])	54
3.10	Grafico con la distribuzione dei mezzi di trasporto del dataset utilizzato per individuare il classificatore migliore.	63
4.1	Schema delle chiamate effettuate nel processo non parametrico di riconoscimento.	68
4.2	Schema descrivente il primo passaggio del processo di clustering.	69
4.3	Esempio di esito delle esecuzioni del GDPMM.	70
4.4	Esempio di parole artificiali generate a partire dall'esito del DPMM.	71
4.5	Schema descrivente il secondo passaggio del processo di clustering.	72
4.6	Esempio di topic mixture generate a partire dall'esito dell'HDP.	73
4.7	Schema descrivente il terzo passaggio del processo di clustering.	73
4.8	Schema descrivente il quarto passaggio del processo di clustering.	74
4.9	Schema descrivente il quinto passaggio del processo di clustering.	75
4.10	Esempio di rapporto finale sui risultati ottenuti dal processo di clustering.	76
4.11	Esempio di dati raggruppati tradotti con etichette reali secondo l'approssimazione descritta.	77
4.12	Esempio di grafico con gli identificatori dei mezzi di trasporto individuati dal processo di clustering per ciascun dato grezzo.	86
4.13	Esempio di grafico contenente l'evoluzione temporale dei mezzi di trasporto veramente utilizzati per ciascun dato grezzo.	87
4.14	Esempio di grafico costruito per confrontare quanto dedotto dal processo di clustering con quanto è successo nella realtà.	88
4.15	Terza finestra dell'interfaccia grafica del software desktop ed esempi di grafici relativi agli assegnamenti delle esecuzioni del DPMM.	89
4.16	Esempio di matrice di similarità costruita da Affinity Propagation.	90
4.17	Scambio di messaggi nell'Affinity Propagation - (Fonte: Science Magazine)	90
4.18	Schema descrivente il funzionamento di Affinity Propagation - (Fonte: Science Magazine)	91
5.1	Grafico con la distribuzione dei mezzi di trasporto per il dataset GL1.	97
5.2	Grafico con l'esito del DPMM per il dataset GL1.	97
5.3	Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset GL1.	98

5.4	Grafico con la distribuzione dei mezzi di trasporto per il dataset GL2.	102
5.5	Grafico con l'esito del DPMM per il dataset GL2.	102
5.6	Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset GL2.	103
5.7	Grafico con la distribuzione dei mezzi di trasporto per il dataset GL3.	107
5.8	Grafico con l'esito del DPMM per il dataset GL3.	107
5.9	Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset GL3.	108
5.10	Grafico con la distribuzione dei mezzi di trasporto per il dataset TW1.	111
5.11	Grafico con l'esito del DPMM per il dataset TW1.	111
5.12	Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset TW1.	112
5.13	Grafico con la distribuzione dei mezzi di trasporto per il dataset TW4.	116
5.14	Grafico con l'esito del DPMM per il dataset TW4.	116
5.15	Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset TW4.	117
5.16	Grafico con la distribuzione dei mezzi di trasporto per il dataset TW5.	122
5.17	Grafico con l'esito del DPMM per il dataset TW5.	122
5.18	Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset TW5.	123
5.19	Grafico con la distribuzione dei mezzi di trasporto per dataset IH8.	129
5.20	Grafico con l'esito del DPMM per il dataset IH8.	129
5.21	Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset IH8.	130
5.22	Grafico con la distribuzione dei mezzi di trasporto per il dataset IH1 con il feature set otto.	132
5.23	Grafico con l'esito del DPMM per il dataset IH1 con il feature set otto.	132
5.24	Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset IH1 con il feature set otto.	133

Elenco delle tabelle

3.1	Elenco dei feature set prodotti al fine di individuare i classificatori migliori mediante Weka.	58
3.2	Descrizione dei dataset utilizzati nella fase di test del software. .	60
3.3	Performance dei classificatori per ciascun feature set.	61
3.4	Performance dei migliori classificatori per ciascun dataset. . . .	62
4.1	Performance del processo di clustering per ciascun dataset analizzato.	85

Capitolo 1

Introduzione

In questo capitolo vengono presentati gli aspetti generali del lavoro. Nella sezione 1.1 viene presentato l'ambito in cui esso si colloca, e nella 1.2 l'obiettivo finale da raggiungere. Nella sezione 1.3 vengono presentate le tecnologie impiegate per ottenere i dati di partenza ed alcuni lavori simili sull'argomento. Si procede poi con la sezione 1.4 dove vengono descritte le tecnologie utilizzate nell'implementazione del sistema software. Altri lavori sull'argomento vengono presentati nella 1.5, per finire poi con la 1.6 dove si può leggere una sinossi dei capitoli successivi.

1.1 Ambito

Negli ultimi anni, la diffusione dei dispositivi mobili ha raggiunto livelli molto elevati, (figura 1.1) indipendentemente da fattori quali sesso ed età, (figure 1.2 e 1.3) e la tecnologia all'interno di essi è sempre più sofisticata. Moltissime attività che prima potevano essere svolte solamente su pc, vengono eseguite con pochi tocchi delle dita su tablet, smartphone e phablet. Gli smartphone sono dispositivi dotati di sistemi operativi avanzati, touchscreen e supporto al software di terze parti; sono i successori dei multimedia phone, caratterizzati dalla presenza di funzioni multimediali di base e di una tastiera qwerty, a loro volta successori dei feature phone, che fornivano solamente chiamate vocali e sms. I tablet si possono considerare dei pc portatili, di dimensioni piuttosto compatte, che consentono l'interazione mediante il tocco della dita, mentre nella categoria dei phablet vengono collocati quei dispositivi troppo grandi per essere smartphone ma troppo piccoli per essere tablet.

Le ultime generazioni di tali dispositivi mobili racchiudono al loro interno diversi tipi di sensori, con i quali è possibile ricavare una grandissima quantità di informazioni riguardanti gli utenti; la presenza di tali sensori, unita alle

capacità computazionali, all'abilità di inviare e ricevere dati di diverso tipo, e all'uso continuo nelle attività giornaliere dei dispositivi "smart" che li contengono, forniscono nuove opportunità di sviluppo per i settori del *data mining*, che consiste nell'estrazione di conoscenza a partire da un insieme di dati, e del *machine learning*, che consiste nello studio e nella costruzione di algoritmi in grado di apprendere dai dati osservati.

In questi ambiti, particolarmente interessanti risultano essere i dati che consentono di studiare la mobilità delle persone, in quanto consentono di capire le abitudini e lo stile di vita, in modo da poter offrire nuove applicazioni, servizi e possibilità di interazione con i dispositivi.

Entrando ancora più nel dettaglio, un task molto importante è quello del riconoscimento dell'attività svolta dall'utente del dispositivo in un determinato momento; le possibili applicazioni offerte da questo compito sono svariate; un esempio è la personalizzazione del comportamento del dispositivo in base alla tipologia di attività riconosciuta (se l'utente sta viaggiando in macchina, un'eventuale chiamata viene riprodotta con il vivavoce attivo), oppure la generazione di un rapporto sullo stato di salute dell'utente stesso, per poter eventualmente consigliare lo svolgimento di attività fisica.

1.2 Obiettivo

L'obiettivo del lavoro consiste nell'approfondire il task del riconoscimento dell'attività svolta dall'utente, cercando di individuare il mezzo di trasporto utilizzato durante un determinato periodo temporale. Il punto di partenza sono i dati registrati dai sensori di un dispositivo mobile, resi disponibili grazie ad un'applicazione installata sul terminale stesso; i sensori in questione sono GPS, accelerometro, giroscopio e magnetometro. Il passaggio successivo consiste nello sviluppo di un software desktop in grado di analizzare tali dati utilizzando diverse tecniche appartenenti al campo del data mining ai fini di individuare quella migliore per eseguire il riconoscimento.

Un primo approccio per eseguire il riconoscimento potrebbe essere di tipo parametrico, in cui si chiede all'utente di stabilire delle soglie per discriminare i mezzi di trasporto (se la velocità è inferiore a 35 km/h allora è in bicicletta, ad esempio); si tratta di un procedimento piuttosto semplice da implementare, tuttavia ci sono due punti critici: l'utente si ritrova a dover specificare un grande numero di soglie per regolare il funzionamento del sistema, e quelle che ritiene giuste potrebbero non esserlo per altri utenti, con un approccio inefficiente di tipo "trial and error"; si ottiene, dunque, una soluzione troppo dipendente dall'utente stesso, mentre sarebbe auspicabile un intervento dello sviluppatore che renda il riconoscimento trasparente e standard rispetto alle

decisioni altrui. Per i motivi precedentemente esposti, si è deciso di implementare un sistema di riconoscimento non parametrico, in modo da rendere non necessario specificare alcun tipo di soglia o il numero di mezzi utilizzato durante il periodo di registrazione. Per eseguire il riconoscimento, in generale, ci sono due tecniche diverse, la *classificazione*, che è una tipologia di apprendimento supervisionato, e il *clustering*, tipologia di apprendimento non supervisionato; tramite quest'ultima si può ottenerne la versione non parametrica, ma si è ritenuto opportuno analizzare anche le possibilità offerte dalla prima tecnica. Tali concetti vengono presentati nella sezione 1.3.1 e approfonditi nei capitoli 3 e 4. Per quanto riguarda il clustering, il sistema viene costruito sulla base del lavoro svolto da F. Sun, Y. Yeh, H. Cheng, C. Kuo e M. Griss in [3]. Essi utilizzano i dati provenienti da due sensori, GPS e accelerometro; un ulteriore obiettivo di questo lavoro, dunque, è impiegare anche i dati di giroscopio e magnetometro e verificare se si ottengono miglioramenti nell'attività di riconoscimento stessa, una volta implementato il sistema. Per concludere, è possibile osservare uno schema riassuntivo dei passaggi fondamentali di questo lavoro nella figura 1.4.

1.3 Stato dell'arte

Scopo di questa sezione è presentare sinteticamente i settori dai quali le tecniche utilizzate provengono ed i sensori utilizzati per la raccolta dei dati successivamente modellati nel software. Nella sezione 1.3.1 viene presentato il machine learning, nella 1.3.2 le attività di data mining ed infine nella 1.3.3 il campo della statistica. Nelle sezioni 1.3.4, 1.3.5, 1.3.6 e 1.3.7 vengono presentati i sensori utilizzati, che sono accelerometro, giroscopio, magnetometro e GPS.

1.3.1 Machine Learning

Il machine learning è una vasta area dell'intelligenza artificiale¹ che si occupa della creazione di algoritmi in grado di imparare dai dati osservati. Tali algoritmi agiscono costruendo un modello basato sui dati di input per fare previsioni o prendere decisioni, piuttosto che seguire esplicitamente istruzioni esplicite di programmazione. Ha connessioni molto strette anche con il settore della statistica; da esso e dall'I.A. ottiene metodi, teorie e domini di applicazione delle tecniche sviluppate grazie a tali contributi. Come indicato in [1], si può affermare che un algoritmo ha imparato anche quando è in grado di

¹L'I.A. si può definire, brevemente, come la capacità di un calcolatore di eseguire funzioni e ragionamenti tipici della mente umana.

modificare il suo comportamento in un modo che gli permetta di raggiungere risultati migliori in futuro; si può testare l'apprendimento confrontando il comportamento passato e quello presente. Le due parole chiave, perciò, sono *conoscenza* e *performance*.

I compiti del machine learning si possono suddividere in tre grandi categorie, che dipendono dalla tipologia di input per la costruzione del modello disponibili al sistema:

- *Apprendimento supervisionato*: al sistema vengono presentati un set di dati di input e gli output desiderati, e l'obiettivo è definire delle regole per mappare gli input negli output;
- *Apprendimento non supervisionato*: anche in questo caso il sistema riceve un set di input, tuttavia non ci sono etichette o indicazioni sulla tipologia di input e sull'output desiderato; il sistema dev'essere in grado di trovare da solo una struttura, ricavando infine le regole per la mappatura di input e output.
- *Apprendimento di rinforzo*: il sistema deve apprendere e adattarsi alle mutazioni dell'ambiente in cui è immerso, utilizzando il concetto di ricompensa, al fine di valutarne le proprie prestazioni; tale concetto prende proprio il nome di rinforzo.

Un'altra categorizzazione delle tecniche di machine learning si può effettuare in base agli output di un sistema di tale tipologia:

- *Classificazione*: gli input vengono suddivisi in due o più classi, e il sistema deve costruire un modello per assegnare gli input a una o più classi;
- *Clustering*: gli input vengono suddivisi in gruppi, senza conoscere in anticipo il numero e le caratteristiche di essi;
- *Regressione*: un processo statistico per verificare se ci sono delle relazioni tra le variabili;
- *Stima della densità*: trovare la funzione di densità della distribuzione di probabilità che ha generato gli input, se esiste;
- *Riduzione della dimensione del problema*: semplificare gli input riducendo le variabili da tenere sotto controllo; ha a che fare con due tecniche chiamate feature selection e feature extraction.

Classificazione, regressione, stima della densità e la riduzione della dimensione sono problemi di apprendimento supervisionato, il clustering ricade nell'apprendimento non supervisionato. Alcune di queste tecniche sono fondamentali nel task del riconoscimento delle attività, come indicato nella sezione 1.2.

1.3.2 Data Mining

Il data mining e il machine learning sono due settori strettamente collegati, molto spesso usano gli stessi metodi e si sovrappongono significativamente. Tuttavia, c'è una differenza fondamentale, che riguarda lo scopo finale di essi. I due concetti, perciò, sono piuttosto sfumati; il data mining trae ispirazione dalle tecniche di machine learning e viene eseguito da una persona immersa in una specifica situazione, che lavora un particolare dataset, con un obiettivo finale in mente, legato alla verifica di ipotesi fatte in precedenza sulla natura dei dati stessi, o per portare alla luce nuova conoscenza sulla struttura di essi. Tipicamente, questa persona vuole testare le performance delle diverse tecniche² per il riconoscimento di pattern strutturali che possono esistere nel dataset. Molto spesso esso è grande, complicato, o può avere problemi speciali (per esempio avere più variabili che rilevazioni). Solitamente, il riconoscimento di questi pattern porta ad alcune intuizioni preliminari in un'area ancora poco esplorata, o ad una predizione più accurata delle rilevazioni future; lo scopo, dunque, non è approfondire la conoscenza del processo che ha portato alla generazione di tali rilevazioni, ma riuscire ad ottenere tali intuizioni per giungere alla scoperta di nuove caratteristiche riguardanti il dataset analizzato.

Secondo quanto descritto in [1] un pattern può essere visto come una black-box³, la cui struttura può essere incomprensibile e che, per tale motivo, viene descritto solo per il risultato a cui porta, o come una white-box,⁴ la cui struttura rivela il processo che ha portato alla sua generazione. Ad ogni modo, vengono definiti *strutturati*, qualunque sia la loro natura, perché aiutano a spiegare eventuali legami esistenti nei dati a disposizione, una volta scoperti; le tecniche citate nella sezione precedente servono, dunque, per farli emergere.

Le applicazioni di questo settore non si limitano all'ambito scientifico; per esempio, sono molto utili nelle ricerche di mercato, ai fini di analizzare grossi

²Quelle descritte nella sezione precedente; per tale motivo l'idea del data mining trae ispirazione dal machine learning.

³Tale termine si usa per descrivere un oggetto che è, appunto, descrivibile solamente per come reagisce a determinate situazioni, dove gli "ingranaggi" che regolano il suo funzionamento non sono visibili.

⁴Tale termine si usa per descrivere un oggetto la cui struttura e gli effetti che vengono applicati alle sollecitazioni che riceve in ingresso sono comprensibili.

archivi di dati riguardanti i clienti in modo da rendere più efficaci i processi di marketing.

A questo punto si può comprendere come l'obiettivo di questo lavoro, descritto nella sezione 1.2, si collochi nel settore del data mining; ci sono un dataset e uno scopo finale da soddisfare a partire da esso, usando le tecniche nate con il machine learning, che a loro volta si basano su concetti appartenenti al campo della statistica.

1.3.3 Statistica

Quello della statistica è un sottoinsieme della matematica. Citando [28], la statistica “fornisce concetti e strumenti per evidenziare gli aspetti rilevanti racchiusi nei dati e per quantificare la forza delle conclusioni che si possono dedurre da tale analisi. I dati sperimentali sono creati in circostanze controllate. L'esperimento può essere replicato un numero di volte arbitrario, mantenendo fede ad un determinato protocollo sperimentale. Solitamente, lo studio dei dati raccolti rileva la presenza di una certa variabilità. I dati rappresentano l'informazione disponibile su certe caratteristiche di una popolazione, ovvero l'intera collezione di unità statistiche sulle quali si cerca l'informazione.”

Le tecniche di machine learning, impiegate per i fini del data mining, inglobano quelle statistiche nei processi di apprendimento dall'osservazione dei dati. Tali tecniche, allo stesso tempo, consentono di comprendere più approfonditamente alcuni processi di generazione dei dati, in contrasto con l'idea che sta alla base del data mining stesso; particolarmente critici sono il concetto di *probabilità*, di *e di distribuzione di probabilità*.

Ancora da [28], “la definizione elementare di probabilità di un evento è data dal rapporto tra il numero di casi ad esso favorevoli e il numero di casi possibili, supposti tutti egualmente probabili. Nella maggior parte dei casi non è così, quindi bisogna introdurre il concetto di *fenomeno aleatorio*; è un fenomeno in riferimento al quale le conoscenze inducono a ritenere possibile una pluralità di esiti. Prima di osservarlo, non è possibile stabilire quale degli esiti si realizzerà. La probabilità, dunque, è una misura che associa ad ogni evento possibile del fenomeno osservato un numero reale, che indica la sua possibilità di realizzazione”.

Un teorema molto importante, che viene chiamato in causa nella sezione 3.6, è quello di *Bayes*:

Teorema 1 (di Bayes) *Dato un evento B e una partizione $A_i, \forall i \in I \subseteq \mathbb{N}$ costituita da eventi, si ha che, $\forall i \in I$,*

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{P(B)} \quad (1.1)$$

Tale teorema, dunque, consente di esprimere la probabilità di una classe di eventi al verificarsi di uno in particolare.

Una distribuzione di probabilità è un modello che descrive la variabilità degli eventi di una determinata *variabile casuale*, funzione matematica che indica tutti i possibili risultati dell'esperimento prima che esso sia realizzato. Tali distribuzioni sono caratterizzate da una *funzione di ripartizione* e da una *funzione di densità*⁵; la prima, calcolata in un determinato punto, indica le probabilità cumulate dei punti precedenti, dove ogni punto corrisponde ad un evento, mentre la seconda descrive la probabilità che si realizzi un evento in particolare, tra quelli definiti dalla variabile casuale. Ricapitolando, data una funzione di ripartizione F , una funzione di densità f , una variabile casuale X e una probabilità P :

$$F_X(x) = P(X \leq x) \quad \text{con} \quad F_X : \mathbb{R} \rightarrow [0, 1] \quad (1.2)$$

$$f_X(x) := \begin{cases} P(X = x_i) = p_i & \text{se } x = x_i, \forall i \in I, \\ 0 & \text{altrimenti.} \end{cases} \quad (1.3)$$

se X è discreta,

$$F_X(x) = \int_{-\infty}^x f_X(t) dx, \quad \forall x \in \mathbb{R} \quad (1.4)$$

$$f_X(x) = \frac{d}{dx} F_X(x) \quad (1.5)$$

se X è continua.

Dalla conoscenza della funzione di ripartizione, dunque, si può ricavare quella di densità, e viceversa. Il *supporto* è l'insieme di tutti i possibili valori che una variabile casuale continua può assumere.

A titolo di esempio, si fornisce la definizione di una variabile casuale *Bernoulliana*, che descrive un esperimento aleatorio dove gli esiti possibili sono successo o insuccesso; la variabile si indica in simboli $X \sim \text{Ber}(p)$, con $p \in (0, 1)$ dove il supporto S_x è $(0, 1)$; si ha dunque che:

$$F_X(x) := \begin{cases} 0 & \text{se } x < 0 \\ 1 - p & \text{se } 0 \leq x < 1 \\ 1 & \text{se } x \geq 1 \end{cases} \quad (1.6)$$

⁵La tecnica di machine learning chiamata regressione si occupa di trovare tale funzione di densità.

$$f_X(x) := \begin{cases} p & \text{se } x = 1 \\ 1 - p & \text{se } x = 0 \end{cases} \quad (1.7)$$

In molti casi, gli algoritmi di clustering e i classificatori vengono rappresentati mediante distribuzioni di probabilità, al fine di effettuare le previsioni necessarie; risulta perciò necessario possedere delle conoscenze di base in questo settore.

1.3.4 Accelerometro

Un *accelerometro* è uno strumento costituito da una massa collegata al sistema stesso mediante un vincolo elastico, per esempio una molla. La misura dell'accelerazione è dedotta dalla misura dello spostamento della massa, in virtù della sua inerzia⁶, rispetto ad un dato sistema di riferimento; tali sensori posso effettuare la misurazione su tutti e tre gli assi, collegando la massa a tre molle diverse; nel calcolo sull'asse Z, in particolare, viene sfruttata anche la forza di gravità. Il sensore può misurare anche accelerazioni negative, poiché gli assi di riferimento hanno l'origine nel centro della massa utilizzata. L'unità di misura è m/s^2 .

Negli smartphone non c'è spazio sufficiente per inserire un sensore così strutturato, perciò il concetto di base è stato veicolato in un altro modo, con un grande sforzo di miniaturizzazione; è stato realizzato un chip costituito da un minuscolo involucro con all'interno una massa, senza molle. Tale massa ha una certa flessibilità, caratteristica che consente di valutare gli spostamenti della massa stessa senza dover utilizzare anche una molla; essa, inoltre, è costituita da piccole lamelle mobili, che si muovono tra una serie di lamelle fisse. L'energia elettrica passa attraverso queste lamelle e varia quando si verifica uno spostamento. Valutando questa energia, il sistema è in grado di calcolare l'accelerazione e usarla per i propri scopi. Solitamente, vengono impiegati tre accelerometri orientati all'interno di un dispositivo, uno per asse.

Le applicazioni dell'accelerometro, come descritto in [27], sono diverse:

- *Rilevazione del movimento*: l'uso più semplice del sensore consiste nel determinare se il dispositivo che lo ospita è in moto oppure no, e se lo è, con quale valore di accelerazione.
- *Tilt sensing*: consiste nel misurare l'inclinazione di un corpo, utile per ruotare lo schermo del dispositivo, per esempio;

⁶Genericamente, la resistenza alla variazione di una qualche grandezza nel tempo.

- *Protezione del dispositivo*: è possibile rilevare improvvise cadute del dispositivo; in tal modo, è possibile disattivare componenti interni come per esempio un hard disk, evitando gravi perdite di dati;
- *Stabilizzazione dell'immagine*: il sensore impedisce al dispositivo di catturare l'immagine finché è ancora in movimento; quando è fermo, anche solo per un millisecondo, la cattura può essere eseguita.
- *Gaming*: un uso piuttosto famoso dell'accelerometro è quello impiegato nella realizzazione dei controller della console Nintendo Wii, in grado di rilevare il movimento del braccio dell'utente realizzando così nuovi videogiochi sfruttanti tale tipologia di interazione.

Ai fini del riconoscimento dei mezzi di trasporto utilizzati dall'utente di un'applicazione mobile, l'uso fondamentale tra quelli indicati è il primo; ad esempio, si immagini di guardare il grafico dell'accelerazione, con il valore di essa sull'asse delle ordinate e il tempo sull'asse delle ascisse. Se si notasse un valore piuttosto alto e stabile nel tempo, si potrebbe dedurre che l'utente in quell'intervallo temporale stava utilizzando un'automobile; ma è solamente una tra tutte le possibili conclusioni che si possono trarre.

1.3.5 Giroscopio

Un *giroscopio* è costituito da un disco rotante, il cui asse di rotazione assume un'orientazione prestabilita; esso è inserito all'interno di tre anelli a loro volta rotanti chiamati *gimbals*, la cui funzione è quella di isolare il disco stesso da forze esterne che possono innescare la rotazione e conseguentemente il cambiamento della direzione di orientamento dell'asse. Se ciò accade, gli anelli si riorientano, riportando l'asse nella direzione di partenza, annullando dunque l'effetto della sollecitazione; se il disco non è inserito nei gimbals, l'asse di rotazione semplicemente modificherà la sua orientazione a seconda della forza. Lo scopo finale, perciò, consiste nel mantenere o misurare l'orientazione del dispositivo che ospita il giroscopio stesso. A partire dal modello di base di un giroscopio, sono state create diverse varianti, in grado di veicolare i principi del sensore in uno spazio ridotto come quello di uno smartphone, sottoforma di circuiti integrati.

Lo scopo più comune per il quale il giroscopio utilizzato, dunque, è capire dove si trova il "down", ovvero mantenere l'orientamento rispetto alla forza di gravità terrestre; l'uso concorrente assieme ad un accelerometro permette, ad esempio, di capire quali accelerazioni sono dovute a tale forza e quali no.

Anche per il giroscopio, come descritto in [28], ci sono diverse applicazioni:

- *Sistemi di guida inerziale*: viene utilizzato per mantenere satelliti, telescopi o missili orientati verso un determinato punto;
- *Stabilizzazione*: garantisce stabilità a veicoli volanti come gli elicotteri radiocomandati;
- *Gyrotheodolite*: strumento utilizzato per mantenere la direzione nello scavo di tunnel minerari;
- *Velivoli*: utilizzato come supporto al volo, per esempio mostrando l'inclinazione dell'aeroplano nell'indicatore dell'altitudine, in modo da aiutare le virate.
- *Dispositivi mobili*: consente di verificare l'inclinazione del dispositivo, e di realizzare tipologie avanzate di interazione nei videogiochi e nelle applicazioni.

Ai fini del riconoscimento del mezzo di trasporto può essere interessante analizzare l'orientamento poiché permette di ottenere determinati tipi di suggerimenti; ad esempio, se nell'intervallo temporale analizzato ci sono frequenti variazioni di tale valore, l'utente potrebbe avere il dispositivo in tasca durante una corsa, ed un eventuale grafico dell'evoluzione temporale del valore sarebbe, perciò, ricco di picchi alti e bassi.

1.3.6 Magnetometro

Il *magnetometro* è uno strumento caratterizzato da due funzionalità, che consistono nella misura della magnetizzazione di un materiale e di forza e direzione di un campo magnetico in un punto dello spazio. Ad esempio, alcuni satelliti sono in grado di misurare il campo magnetico terrestre per rilevare eventuali anomalie. Negli ultimi anni, anche questo sensore è stato miniaturizzato sotto forma di circuito integrato e inserito all'interno di dispositivi mobili di vario tipo.

Come indicato in [30], un campo magnetico viene descritto da un vettore, caratterizzato da una direzione nello *spazio vettoriale*⁷ tridimensionale e da un'intensità. L'unità di misura dell'intensità di un campo magnetico è il *Tesla*,

⁷Si tratta di una struttura algebrica che generalizza l'insieme dei vettori di uno spazio n -dimensionale, dotato di due operazioni fondamentali, che sono la somma tra vettori e la moltiplicazione di un vettore per uno scalare appartenente all'insieme dei numeri reali. Il concetto di spazio vettoriale viene utilizzato anche in altri campi, come la grafica 3D interattiva e la ricerca operativa.

nel sistema internazionale, mentre il *Gauss* viene impiegato nel sistema CGS⁸; diecimila *Tesla* corrispondono a un *Gauss*. Il campo magnetico terrestre, per esempio, varia da ventimila nano tesla a ottantamila nano tesla. A volte si parla di *teslametri* o *gaussametri*, a seconda dell'unità utilizzata.

Vi sono due categorie di magnetometri, se per classificarli si considera la tipologia di misura effettuata. I magnetometri *scalari* calcolano la magnitudo del vettore descrivente il campo, mentre i magnetometri *vettoriali* calcolano le tre componenti del vettore nello spazio tridimensionale.

I magnetometri sono classificabili anche in base al modo in cui vengono impiegati; si dicono stazionari se la misura viene effettuata sempre nello stesso punto, mentre quelli portatili sono pensati per essere usati in movimento.

Come per i due sensori trattati precedentemente, anche il magnetometro ha molti usi:

- Archeologia: può essere usato per trovare siti archeologici, relitti sommersi ecc., ma anche per individuare pietre magnetiche come basalto e granito.
- Militare: una rete di magnetometri viene impiegata per monitorare le attività dei sottomarini; paesi come gli Stati Uniti e il Canada possiedono una tecnologia molto avanzata, per tale scopo.
- Trivellazioni: viene utilizzato per fornire informazioni aggiuntive sulla geologia del terreno sottostante, per supportare l'esecuzione delle trivellazioni.
- Aurore: i magnetometri possono fornire informazioni sulle aurore ancora prima che la luce diventi visibile, misurando l'effetto del sole sul campo magnetico terrestre.
- Dispositivi mobili: al giorno d'oggi, il prezzo di un magnetometro miniaturizzato è meno di un dollaro, perciò praticamente ogni dispositivo possiede tale sensore al suo interno; una sua applicazione piuttosto semplice è come bussola, poiché consente di puntare verso il polo nord magnetico.

Per quanto riguarda il riconoscimento dei mezzi di trasporto utilizzati dagli utenti, forse è il sensore più difficile da interpretare. La presenza o meno di campi magnetici potrebbe segnalare che l'utente si trova in luoghi diversi; si pensi genericamente ad uno spostamento in auto; il punto di partenza potrebbe

⁸Tale acronimo sta per centimetre-gram-second ed indica una variante del ben più noto sistema metrico. Il CGS usa le unità che compongono il suo nome come base per la misura di lunghezza, massa e tempo.

essere in città, mentre l'arrivo in campagna; se il campo magnetico riduce notevolmente la sua forza, si può pensare che lo spostamento sia stato piuttosto lungo; se il campo rimane agli stessi livelli del punto di partenza, invece, lo spostamento non è stato così marcato; tale assunzione potrebbe portare ad escludere determinati mezzi. Ad ogni modo, il magnetometro da solo non è in grado di fornire molte informazioni; l'uso in sinergia con il GPS, invece, può essere più significativo, per via delle informazioni che quest'ultimo sensore è in grado di offrire, come si vedrà nella sezione seguente.

1.3.7 GPS

Il *Global Positioning System* (da cui l'acronimo GPS) è un sistema di navigazione satellitare spaziale, in grado di fornire informazioni temporali e relative alla posizione di qualsiasi punto sulla terra; tali informazioni vengono inviate a degli appositi ricevitori GPS, installati all'interno dei dispositivi. Il sistema consiste di trentadue satelliti che orbitano a dodicimila miglia di altezza; essi vengono alimentati dall'energia solare, e possiedono delle batterie di riserva che permettono di proseguire il volo quando non sono raggiunti dalla luce.

L'intera struttura del GPS si può dividere in tre sezioni:

1. *segmento spaziale*: indica tutti i satelliti in orbita;
2. *segmento di controllo*: indica un insieme composto da stazioni di monitoraggio e antenne terrestri del sistema;
3. *segmento utente*: indica tutti i dispositivi civili e militari che possiedono un ricevitore gps al loro interno;

Il governo degli Stati Uniti gestisce i primi due segmenti.

Ogni satellite del GPS compie un giro attorno alla Terra due volte al giorno, viaggiando a velocità pari a settemila miglia orarie e trasmettendo continuamente informazioni a terra sottoforma di segnali. Le orbite dei trentadue satelliti sono tali che ogni punto del pianeta è sempre coperto da almeno nove di essi. I ricevitori GPS utilizzano queste informazioni per effettuare la triangolazione⁹ e calcolare l'esatta posizione dell'utente. Un ricevitore deve avere almeno tre satelliti in vista per calcolare la posizione in due dimensioni e tracciare il movimento, mentre deve poter accedere all'informazione di quattro satelliti per calcolare la posizione nello spazio tridimensionale, latitudine, longitudine e altitudine.

⁹Tecnica che permette di calcolare la distanza tra due punti sfruttando le proprietà dei triangoli. Tale tecnica si basa sulla trigonometria, disciplina matematica che tratta lo studio dei triangoli a partire dai loro angoli.

I segnali satellitari viaggiano seguendo una linea retta dal punto nello spazio in cui vengono trasmessi e sono in grado di attraversare la maggior parte degli oggetti che colpiscono, quali nuvole, vetro e plastica; tuttavia, non sono in grado di oltrepassare le montagne o gli edifici più grossi.

Il segnale trasmesso si compone di tre parti distinte:

1. *codice pseudocasuale*: si tratta di un identificatore numerico necessario per capire quale satellite sta trasmettendo;
2. *dati effemeridi*: sono costituiti dallo status del satellite, dal tempo e dalla posizione corrente; sono fondamentali per determinarne la posizione;
3. *almanacco*: dati che indicano al ricevitore dove si dovrebbero trovare tutti i satelliti della rete in qualsiasi momento della giornata;

Una volta determinata la posizione dell'utente, possono essere calcolate molte informazioni accessorie, quali la distanza del viaggio, la velocità, gli orari di tramonto e alba e la distanza mancante per raggiungere una destinazione, ad esempio.

Nonostante originariamente fosse un progetto militare, le applicazioni di questa tecnologia per scopi anche di tipo civile sono svariate, nonostante in tale ambito esistano delle restrizioni. Per esempio, per quanto riguarda l'uso civile:

- *Veicoli automatici*: impostare la traiettoria di auto senza pilota umano;
- *Geotagging*: Applicare coordinate geografiche ad oggetti digitali come le fotografie, tramite moduli appositi all'interno delle macchina fotografiche;
- *Geofencing*: utilizzare i satelliti per identificare la posizione di un essere umano, di un oggetto, di un veicolo o di un animale;
- *Navigazione*: aiuto alla guida per raggiungere una data destinazione da un punto di partenza.

Mentre in ambito militare le applicazioni sono, per esempio:

- *Tracciamento del bersaglio*: molte tipologie di armamenti utilizzano il gps per tracciare le coordinate del bersaglio da colpire;
- *Ricerca e Soccorso*: utilizzo dei satelliti per localizzare bersagli alleati abbattuti e organizzare l'operazione di recupero.

Per quanto riguarda lo scopo di questo lavoro, i dati forniti dal GPS sono quelli più utili; a partire da latitudine, longitudine, tempo e velocità è possibile calcolare la distanza media percorsa o la velocità media, per esempio; in tal modo si può ottenere un maggior numero di dati utili per le attività di *éclustering* e classificazione. Ad esempio, se dai dati GPS disponibili si nota che sono stati percorsi dieci chilometri in tre minuti, probabilmente l'utente non stava utilizzando una bicicletta; come si vedrà nei capitoli successivi, il riconoscimento è stato effettuato tenendo sempre in considerazione tale sensore, mentre gli altri vengono considerati opzionali, ed utilizzati a seconda dei casi.

1.4 Tecnologie utilizzate

Scopo di questa sezione è presentare brevemente le tecnologie che sono state utilizzate nell'implementazione del software utilizzato in questo lavoro. Nelle sezioni 1.4.1 e 1.4.2, dunque, vengono descritti Java e Java FX, rispettivamente linguaggio di programmazione orientato agli oggetti e linguaggio di scripting derivato da Java stesso utilizzato per la creazione di applicazioni RIA.

1.4.1 Java

Come descritto poco fa, Java è il linguaggio scelto per implementare il software utilizzato per l'attività di analisi dei dati grezzi a disposizione. Si tratta di un linguaggio di programmazione orientato agli oggetti, che consiste in un'evoluzione del paradigma imperativo. Possiede le astrazioni di oggetto¹⁰, classe, metodo e consente di sfruttare tecniche quali ereditarietà, incapsulamento e polimorfismo, in modo da favorire la modularizzazione e il riuso del codice; supporta, inoltre, la programmazione concorrente mediante l'utilizzo dei thread¹¹. Tali caratteristiche sono state sfruttate nell'implementazione. Per quanto riguarda la documentazione è stato utilizzato Javadoc, strumento contenuto all'interno della piattaforma Java stessa; la documentazione viene prodotta in codice HTML ed è esplorabile mediante browser.

Java è un linguaggio semi interpretato, poiché una volta compilato viene tradotto in un codice intermedio, il bytecode. Tale codice, successivamente, viene interpretato ed eseguito dalla JVM¹², che può essere installata su qual-

¹⁰Entità in grado di modificare il loro stato, descritto da attributi, mediante lo scambio di messaggi.

¹¹Una definizione semplicistica di thread è dividere un processo in diversi sottoprocessi; avere diversi thread in un processo, perciò, è come avere diversi processi in esecuzione su un calcolatore.

¹²Java Virtual Machine, componente della piattaforma Java che esegue i programmi tradotti in bytecode dopo una prima compilazione

siasi sistema, Windows, Linux o Unix. Uno dei punti di forza di Java, dunque, è la portabilità, motivo che lo rende uno dei linguaggi più usati al mondo. Per via della sua diffusione, nel corso degli anni sono state sviluppate centinaia di librerie che consentono di risolvere moltissimi problemi implementativi, come, ad esempio, gestione dei file, interazione con le basi di dati relazionali, e quant'altro. Lo svantaggio è che l'esecuzione di un programma Java è molto più lenta di un programma C++, per esempio, per via del passaggio al codice intermedio interpretato dalla JVM.

1.4.2 Java FX

Java FX, citando [16], consiste in: “un set di pacchetti di contenuti grafici e multimediali che permettono agli sviluppatori di progettare, creare, testare e pubblicare applicazioni RIA¹³ che operano in modo consistente su diverse piattaforme”. Il codice Java FX fa parte delle API Java; in tal modo può essere richiamato all'interno di qualsiasi classe, a patto di avere la settima versione del JDK¹⁴. L'aspetto delle applicazioni che sfruttano questo codice può essere personalizzato tramite fogli di stile CSS¹⁵ (si veda [18]), separandolo, in questo modo, dalla logica dell'applicazione stessa. Qualora si volesse ottenere una separazione ancora più netta tra questi due aspetti, si può scrivere l'interfaccia utente in FXML, un linguaggio di scripting basato su XML;

Le applicazioni Java che sfruttano il pacchetto Swing per l'interfaccia utente possono utilizzare anche gli elementi grafici di Java FX inserendoli all'interno di contenitori speciali. Il software costruito per questo lavoro, infatti, si affida proprio a Swing per l'interfaccia utente, mentre di Java FX vengono sfruttate le funzionalità per la produzione di grafici di vario tipo; a tale scopo, vengono utilizzate le informazioni contenute in [19], [20] e [21]. Alcuni particolari dell'interfaccia vengono, infine, personalizzati tramite CSS. L'interoperabilità tra i due sistemi viene garantita poiché, nonostante Java FX sia il futuro, Swing rimane una tecnologia “rock solid”, in grado di funzionare in qualsiasi calcolatore, anche se la versione di Java installata risulta essere superata.

¹³Tale acronimo sta per Rich Internet Application, ovvero applicazioni web che possiedono le caratteristiche di applicazioni desktop, senza eseguire un'installazione su disco fisso. Si caratterizzano per l'interattività, la multimedialità e la velocità di risposta agli input dell'utente attraverso l'interfaccia. Le RIA, dunque, si basano su un'architettura distribuita client-server.

¹⁴Java Development Kit, ovvero l'insieme di tutti gli strumenti per programmare in Java.

¹⁵Cascading Style Sheet, linguaggio per definire l'aspetto di documenti HTML, XML e XHTML.

1.5 Altri lavori

In questa sezione vengono brevemente illustrati lavori nell'ambito del riconoscimento delle attività, svolti da ricercatori; lo studio di tali lavori è stato fondamentale per la preparazione allo svolgimento di questo. I procedimenti seguiti risulteranno più chiari una volta terminata la lettura dei capitoli successivi.

F. Sun, Y. Yeh, H. Cheng, C. Kuo e M. Griss in [3] propongono un framework non parametrico per scoprire le routine degli utenti senza conoscere il numero. Le loro analisi sono svolte su due dataset pubblici, il primo contenente dati riguardo alle attività svolte in trentaquattro giorni, mentre il secondo contiene dati relativi ai trasporti. Inizialmente vengono descritti i vantaggi di un metodo non parametrico rispetto ad uno parametrico; successivamente, viene descritto il processo di estrazione delle feature dai dati di accelerometro e GPS, poi mappate in parole artificiali utilizzando un algoritmo di clustering chiamato DPMM; tali parole artificiali, infine, vengono sottoposte all'azione di un ulteriore algoritmo di clustering, chiamato HDP, e le routine vengono così scoperte. Vengono, infine, proposte delle metriche per un'analisi quantitativa e qualitativa dei risultati ottenuti. Si tratta del lavoro più importante tra quelli studiati, sul quale si basa l'implementazione realizzata in questo lavoro per l'analisi dei dati provenienti dai sensori descritti nelle sezioni precedenti.

J. Kwapisz, G. Weiss e S. Moore in [4] illustrano il loro approccio al riconoscimento delle attività, utilizzando solamente i dati provenienti dall'accelerometro; il dataset utilizzato è costituito dalla rilevazioni effettuate da diversi utenti mediante un'applicazione Android, incaricati di eseguire alcuni tipi di attività trasportando il dispositivo. Anche in questo caso vi è una fase di trattamento dei dati grezzi, seguita dall'estrazione di feature. Quest'ultime, infine, vengono testati alcuni classificatori, per descriverne il livello di accuratezza nell'individuazione delle attività a seguito del trattamento dei dati grezzi etichettati.

A. Rasekh, C. Chen e Y. Lu in [5] propongono un'ulteriore procedura di riconoscimento delle attività a partire dai dati forniti dall'accelerometro; analogamente ai lavori precedenti, gli utenti di un'applicazione svolgono l'attività di registrazione dei dati; una prima differenza consiste nell'utilizzo di un algoritmo di feature extraction chiamato LDA per ridurre la dimensione dei dati di partenza; solo a questo punto vengono estratte le feature finali. Una

seconda differenza rispetto al lavoro precedente è che sono state testate altre tipologie di classificatori.

M. Lin e W. Hsu in [2] svolgono un lavoro di più ampio spettro, riguardo all'utilizzo dei dati GPS per estrarre pattern riguardanti la mobilità degli utenti. Descrivono inizialmente strategie per fare inferenza relativamente ai luoghi significativi per gli utenti, e per predirne gli spostamenti. Successivamente discutono vari approcci parametrici per analizzare i mezzi di trasporto utilizzati, indicando le feature più adatte a tale scopo ed evidenziando i punti comuni a ciascun approccio. Si prosegue analizzando i diversi modi per eseguire l'analisi delle traiettorie di spostamento degli utenti, illustrando i diversi approcci possibili per lo svolgimento di tale compito, che consistono nel clustering, nell'estrazione delle traiettorie basandosi sui grafi e nel trattamento di esse come oggetti spazio-temporali. Gli autori procedono proponendo metodi per riconoscere le attività dipendenti da determinati luoghi, sfruttando, dunque, anche dati relativi al posizionamento dell'utente e non solo al movimento di esso. Viene, infine, fornita una panoramica su diversi problemi contingenti ancora da affrontare. Il lavoro svolto da questi ricercatori è stato utile per la sua caratteristica di essere una panoramica generale su diversi aspetti del riconoscimento delle attività.

Y. Zheng, Q. Li, Y. Chen, X. Xie, W. Ma in [32] descrivono uno dei primi approcci per lo studio della mobilità basandosi sui dati GPS. In particolare, indicano l'individuazione di un set di feature particolarmente robusto alle condizioni del traffico, per poi proporre un algoritmo basato sui grafi per realizzare la classificazione dei dati. Anche in questo caso si ha a che fare con un apprendimento supervisionato. Il dataset è costituito dalle rilevazioni GPS di sessantacinque utenti in un periodo di dieci mesi. Si tratta del lavoro più vecchio tra quelli in bibliografia, e i risultati sono stati sfruttati in [2] nello sviluppo di un apprendimento non supervisionato.

Come si può notare la maggior parte dei lavori analizzati si concentrano su approcci supervisionati basati su classificatori e dataset etichettati, per svolgere l'attività di riconoscimento; gli approcci non supervisionati basati su algoritmi non parametrici sono un settore esplorato solamente negli ultimi tempi.

1.6 Sinossi

Nel capitolo 2 viene descritto il dataset utilizzato per l'analisi e le operazioni preparatorie da effettuare sui dati grezzi prima del riconoscimento.

Nel capitolo 3 viene illustrato il processo di estrazione delle feature sottoposte alle diverse tipologie di classificatori implementati all'interno di Weka, software che consente di testare molti algoritmi di machine learning, al fine di eseguire una panoramica dei diversi modi per mettere in pratica l'apprendimento supervisionato, basato sull'addestramento del classificatore stesso. Il capitolo si chiude con una sezione di analisi dei risultati ottenuti.

Nel capitolo 4 viene descritto l'approccio non parametrico per riconoscimento del mezzo di trasporto utilizzato dagli utenti. Vengono inizialmente presentate le definizioni di base necessarie a comprenderne le caratteristiche; si descrive, successivamente, la preparazione dei dati da sottoporre al DPMM, algoritmo di clustering non parametrico scelto ed implementato nel software. Successivamente, si ottengono le cosiddette parole artificiali a partire dai dati grezzi raggruppati nei vari cluster; tali parole artificiali vengono fornite in input ad un ulteriore algoritmo di clustering. Il capitolo si avvia alla conclusione con la descrizione del terzo ed ultimo algoritmo, utilizzato sui risultati del passo precedente, chiamato Affinity Propagation, al fine di individuare i mezzi di trasporto. Anche in questo caso il capitolo si chiude con un'analisi dei risultati ottenuti.

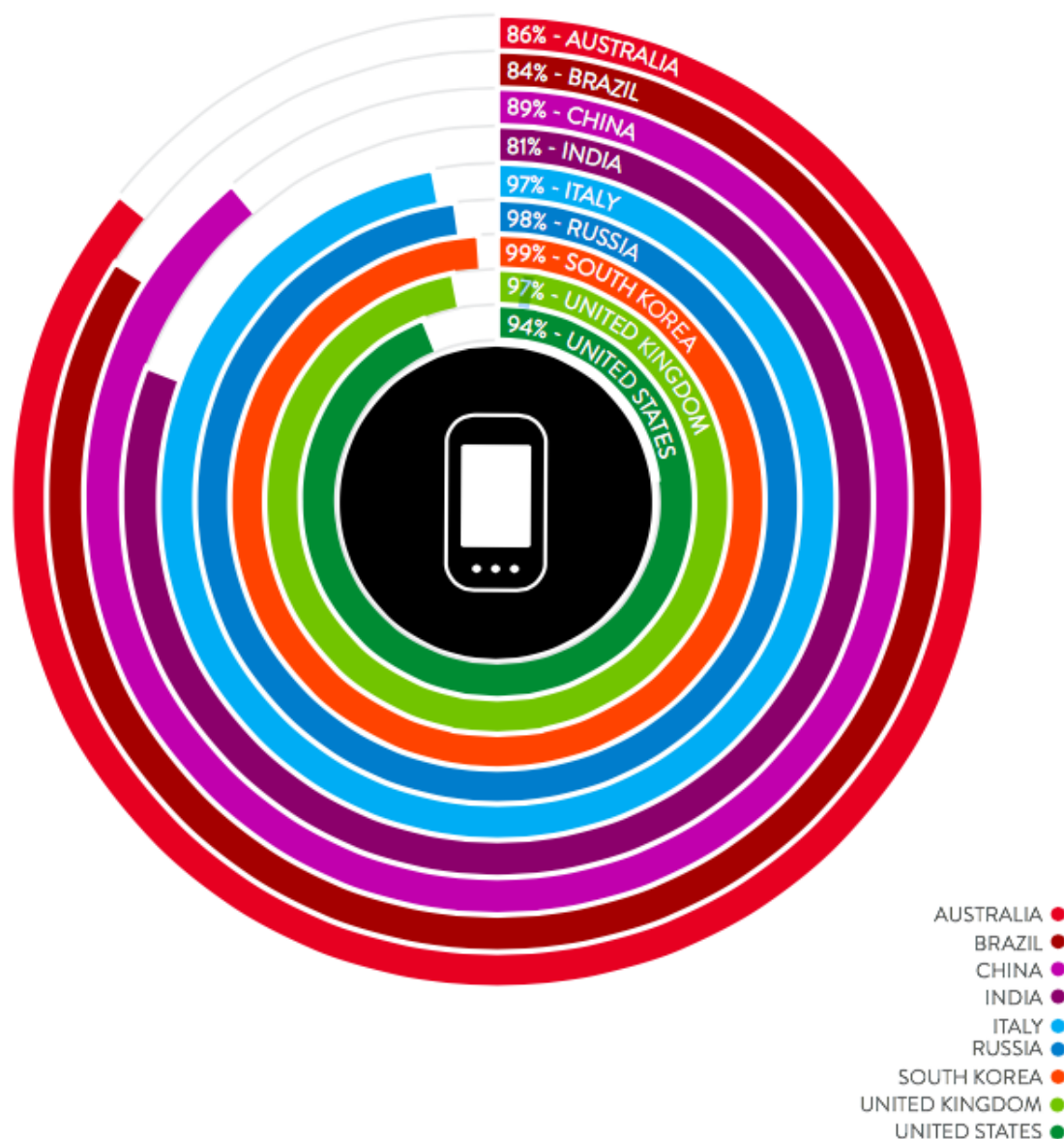


Figura 1.1: Penetrazione degli smartphone calcolata negli stati più tecnologicamente avanzati del pianeta – Febbraio 2013 (Fonte: Nielsen Global Smartphone Insights)

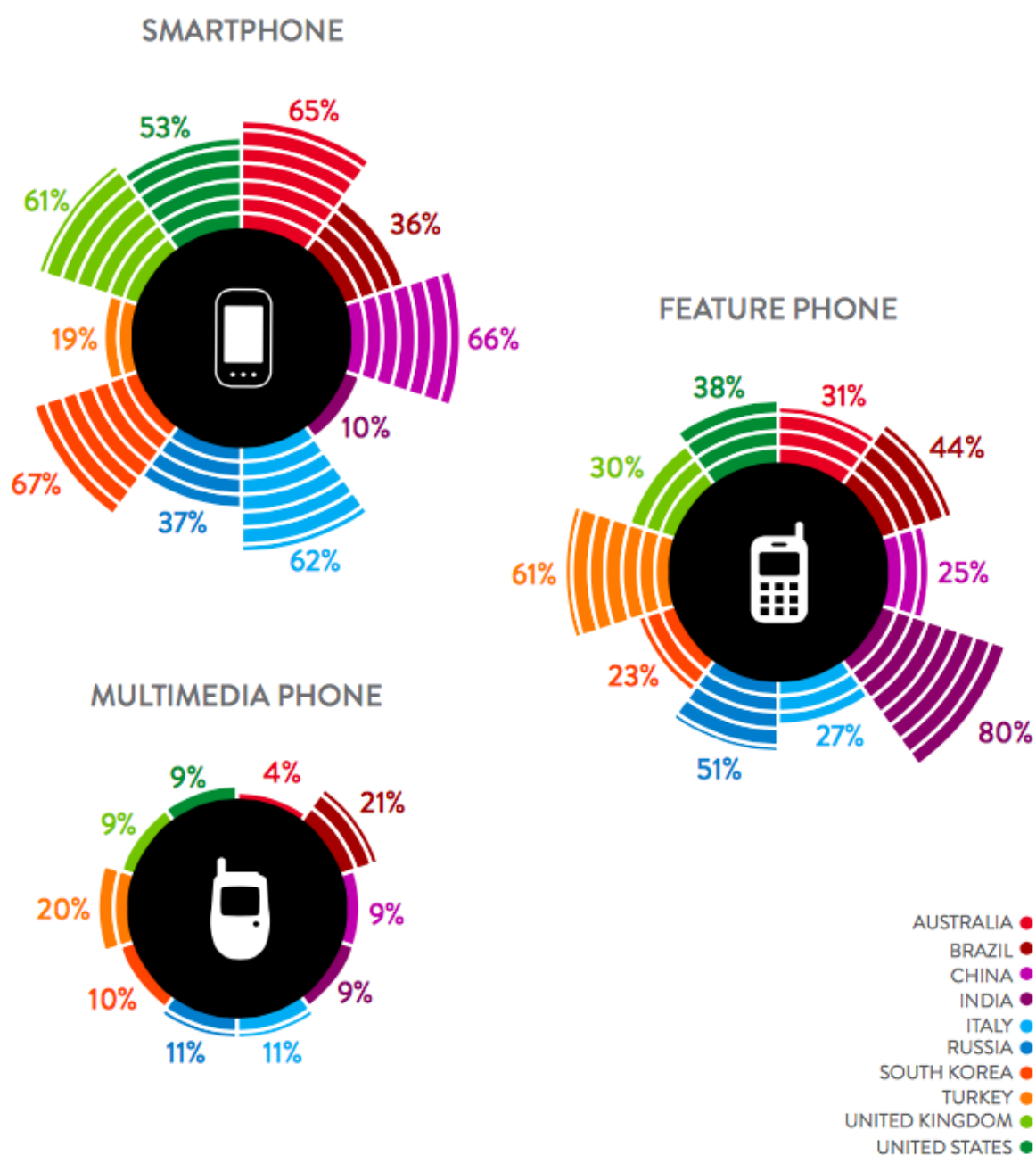


Figura 1.2: Percentuali di utilizzo delle tre tipologie di cellulare – Febbraio 2013 (Fonte: Nielsen Global Smartphone Insights)

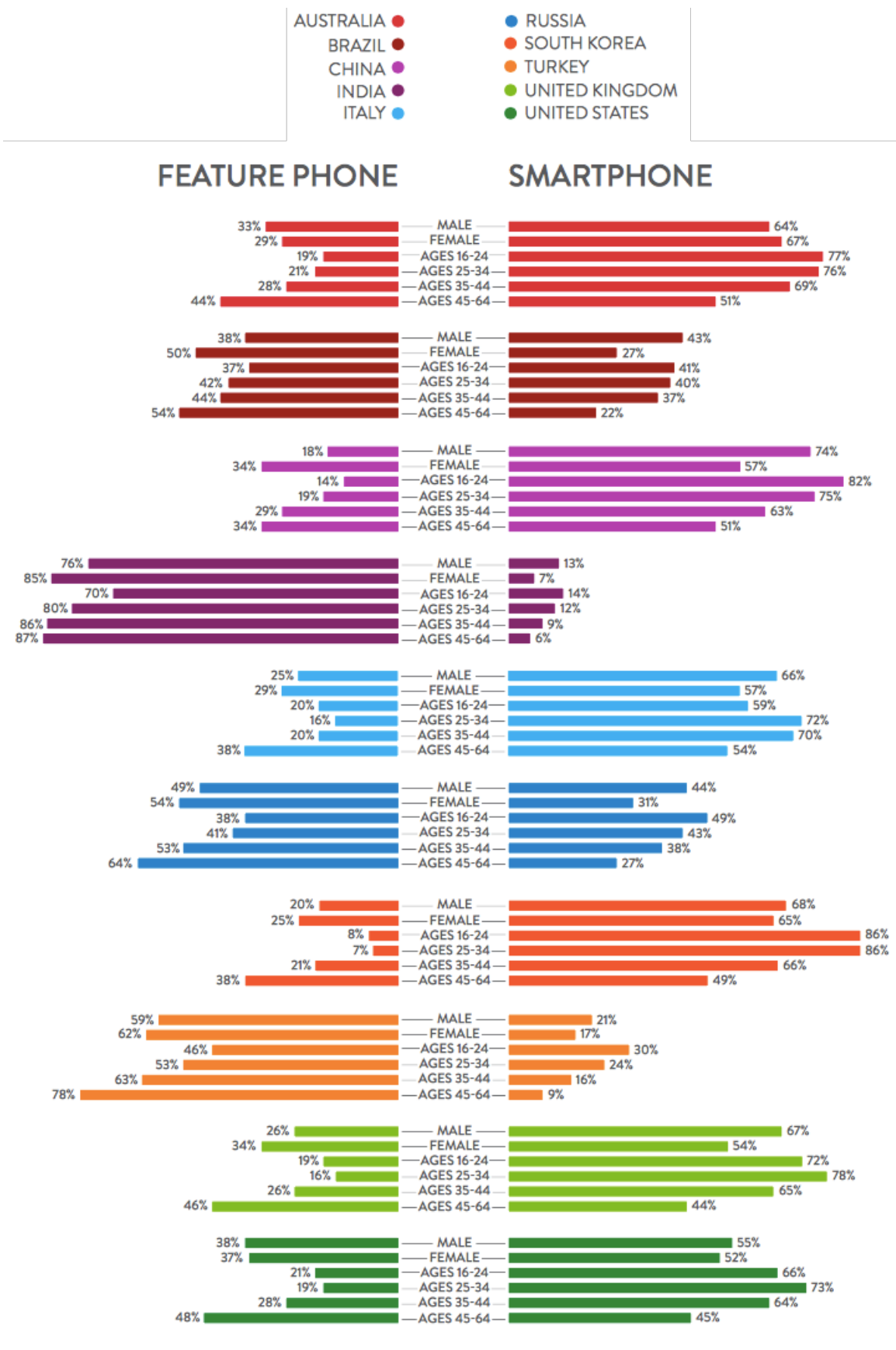


Figura 1.3: Percentuali di utilizzo di smartphone e feature phone suddivise per sesso ed età – Febbraio 2013 (Fonte: Nielsen Global Smartphone Insights)

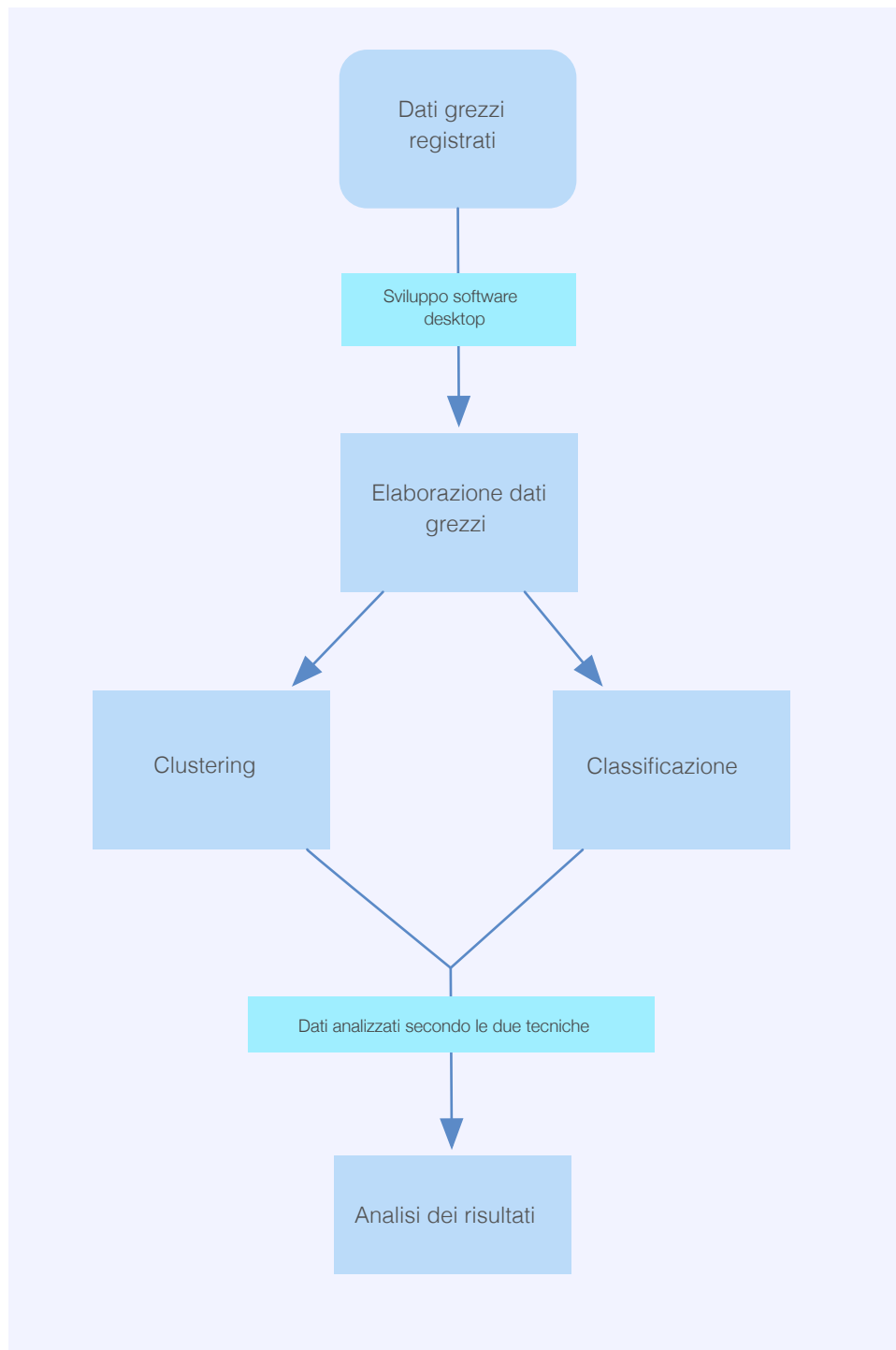


Figura 1.4: Passaggi fondamentali nella realizzazione del lavoro proposto.

Capitolo 2

Elaborazione dei dati grezzi

Scopo di questo capitolo è presentare la struttura del dataset utilizzato e le operazioni preliminari di estrazione e preparazione dei dati grezzi necessarie per le attività di clustering e classificazione.

2.1 Dataset utilizzato

Il dataset utilizzato è stato creato grazie a *Lifetracker*, un'applicazione per smartphone. Il compito svolto da essa consiste nell'interfacciarsi con i quattro sensori descritti nel capitolo 1, presenti all'interno del dispositivo nel quale avviene l'installazione, e registrarne i dati su file di testo, ad intervalli regolari. Un file di testo prodotto dall'applicazione corrisponde ad una particolare sessione di registrazione effettuata. Un dato grezzo, perciò, è costituito dall'insieme dei valori contenuti in una riga di un file prodotto dall'applicazione, istante per istante. I valori di ciascuna riga sono strutturati nel modo seguente:

```
latitude,longitude,altitude,speed,course,horizontalAccuracy,  
accelX_accelY_accelZ,gyroX_gyroY_gyroZ,magnetX_magnetY_magnetZ,  
brightness,proximity,timestamp,movement,trackingType
```

Nella figura 2.1 si può osservare un esempio concreto di dati grezzi formati in una riga.

```
45.780542,12.823671,3.777554,0.000000,177.890625,5.000000,-0.054916_-0.72908  
0_-0.637589,-0.376248_-3.430337_-0.061272,-27.170410_-289.412781_-52.712997,  
1.000000,0,2014-05-30_18:20:13,r,f
```

Figura 2.1: Riga d'esempio tratta da un file del dataset utilizzato.

Il penultimo dato, in particolare, indica il mezzo di trasporto utilizzato in quel determinato momento; i tester dell'applicazione, infatti, avevano il compito di fornire tale informazione durante la sessione d'uso. In tal modo, è stato possibile ottenere dati etichettati necessari per i classificatori, come si vedrà nel capitolo seguente. Per quanto riguarda il clustering, e, dunque, l'approccio non parametrico, tale informazione non è necessaria; tuttavia, rimane utile per verificare la correttezza del processo.

2.2 Preparazione dei dati

Come si è visto, i dati sono archiviati in file di testo, riga per riga, con i valori formattati secondo un certo criterio. Sono necessarie, dunque, delle operazioni preliminari prima di fornirli in input alle fasi successive dell'analisi. All'avvio, il software si presenta con la prima finestra dell'interfaccia grafica, come si può vedere nella figura 2.2. Si può indicare al programma un file singolo da elaborare, oppure un'intera cartella, a patto che i file contenuti in essa siano formattati nel modo descritto nella sezione 2.1. I parametri che si possono osservare nella parte inferiore dell'interfaccia servono per la classificazione; se ne parlerà approfonditamente nella sezione 3.2.

Ogni sensore viene modellato nel programma da una classe, che si occuperà di memorizzare in una struttura dati i valori di interesse per il sensore modellato e le informazioni relative al tempo e all'etichetta, analizzando ciascun dato grezzo. Nella figura 2.3 si può osservare la gerarchia delle classi che modellano i quattro sensori; i metodi ereditati da tali classi sono operazioni eseguite nello stesso modo da ciascun sensore.

Concretamente, vengono istanziati oggetti per ciascun sensore utilizzato e vengono eseguite le chiamate ai metodi indicati in figura 2.4; due rami separati dell'albero indicano che tali chiamate possono essere eseguite in qualsiasi ordine.

loadRawData è il metodo che si occupa di caricare in una struttura dati i valori di interesse per il sensore che si sta utilizzando. Un dato grezzo contiene tutti i valori di ciascun sensore registrati in un determinato istante temporale; un oggetto della classe accelerometro, quindi, è interessato alla tripla (accelX, accelY, accelZ), un oggetto della classe giroscopio alla tripla (gyroX, gyroY, gyroZ) e infine un oggetto della classe magnetometro a (magnetX, magnetY, magnetZ); tutte queste triple descrivono i tre vettori tridimensionali che rappresentano l'accelerazione, l'inclinazione e direzione del campo magnetico rilevate nei tre assi. La classe Speed, che corrisponde al GPS, invece, è interessata alla tripla (speed, latitude, longitude). Per estrarre tali informazioni, il metodo va ad eseguire l'operazione di splitting della riga, che consiste

nel suddividere la riga stessa in più porzioni secondo un carattere separatore, in questo caso la virgola; tali valori vengono restituiti come vettore di stringhe, rendendo possibile la separazione della loro elaborazione. Il metodo è sovraccarico¹ e riceve come parametri un primo carattere separatore, la posizione in cui si trova il dato d'interesse una volta suddivisa la riga secondo il primo carattere ed un eventuale secondo carattere separatore; quest'ultimo serve ad accelerometro, magnetometro e giroscopio per effettuare un ulteriore splitting al fine di dividere in tre parti la tripla contenente i valori per ciascun asse. I valori ottenuti, infine, vengono memorizzati in apposite liste.

loadTimeStampData e *loadActivityData* vanno a recuperare le informazioni riguardanti il tempo e l'etichetta che l'utente ha indicato tramite interfaccia; funzionano in modo analogo a *loadRawData* e le uniche differenze sono che il primo metodo riceve come parametri, oltre al primo carattere separatore e alla prima posizione, caratteri e posizioni aggiuntivi per dividere in due parti i valori relativi alla data e ora, a loro volta divise in tre parti ciascuna, mentre il secondo metodo riceve come parametri solamente il primo carattere e la prima posizione.

Un ulteriore passaggio, che viene svolto solo per accelerometro, magnetometro e giroscopio, è la chiamata di *computeMagnitudeData*, metodo che si occupa del calcolo della *magnitudo*, da effettuarsi rigorosamente dopo quella a *loadRawData*. La *magnitudo* si può descrivere come la dimensione di un oggetto matematico; tale proprietà consente di verificare se tale oggetto è più o meno grande di uno dello stesso tipo, definendo, dunque, un ordinamento. Per gli obiettivi di questo lavoro, è poco interessante analizzare le singole componenti del vettore; la *magnitudo* è utile proprio perché consente di capire quando accelerazione, inclinazione o campo magnetico sono più o meno "intensivi", senza entrare nel merito delle singole componenti. Il lavoro svolto dal metodo, quindi, consiste nel creare un array dove memorizzare, per ogni tripla di ciascun sensore, la relativa *magnitudo*, calcolata secondo la seguente formula:

$$Mag(v) = \sqrt{u_1^2 + u_2^2 + u_3^2} \quad \text{con } v=(u_1, u_2, u_3) \quad (2.1)$$

Una volta eseguite tutte queste operazioni i dati grezzi sono memorizzati in strutture dati apposite ed è possibile procedere con gli stadi successivi dell'elaborazione; come ultimo passaggio, le *magnitudo* di accelerometro, giroscopio e magnetometro, e il valore relativo alla velocità per quanto riguarda il GPS, vengono visualizzati in grafici; se è stata elaborata un'intera cartella, è possibile osservarli per ciascun file contenuto in essa. Al termine dell'elaborazione,

¹Un metodo si dice sovraccarico se sono presenti diverse sue implementazioni dove la firma è la stessa ma i parametri sono in diverso numero o ordine.

infatti, l'interfaccia utente viene aggiornata con l'aggiunta dei grafici stessi, mentre nella sezione inferiore di essa vengono aggiunti dei pulsanti per visualizzare le componenti dei dati grezzi in formato tabellare e per salvare i dati stessi nel formato ARFF (di cui si parla nella sezione 3.3). A titolo d'esempio, le figure 2.5, 2.6, 2.7 e 2.8 corrispondono ai grafici dei dati elaborati per tutti e quattro i sensori, provenienti da un file tratto dal dataset; in tale sessione di registrazione sono state svolte due attività diverse.

In aggiunta al dataset prodotto grazie a Lifetracker, si è deciso di sfruttare due ulteriori dataset resi disponibili al pubblico da attori esterni al fine di verificare che il sistema sviluppato sia in grado di funzionare con dati provenienti da altre fonti. Eseguire tale operazione è interessante anche perché tali dati sono stati raccolti secondo modalità diverse di campionamento dei singoli sensori e contengono attività non registrate dall'applicazione utilizzata. Sono potute emergere, dunque, alcune delle considerazioni descritte nelle sezioni di analisi dei risultati di questo lavoro. Il primo dataset pubblico utilizzato è quello del progetto *Geolife* ed è disponibile in [42]; è costituito solamente da dati GPS relativi alle traiettorie (ovvero latitudine, longitudine ed altitudine) provenienti da centottantadue utenti, raccolti tra il 2007 e il 2012 in Asia da Microsoft Research. Il secondo dataset pubblico (si veda [43]) è stato raccolto dall'università di Twente, in Olanda, e consiste nei dati relativi ad accelerometro, magnetometro e giroscopio raccolti da alcuni utenti mediante dispositivi mobili, come nel caso di Lifetracker; due aspetti interessanti di questa raccolta sono l'elevata frequenza di campionamento (cinquanta campioni per secondo) ed il fatto che sono stati raccolti contemporaneamente i dati dei sensori precedentemente nominati da dispositivi posizionati, in particolare, su un braccio, sulla cintura dei pantaloni, in tasca e sul polso degli utenti stessi. Per sfruttare questi due dataset con il sistema sviluppato si è resa necessaria un'operazione di parsing per poter riscrivere i dati nel formato richiesto dal sistema stesso. L'utente del software, tramite la finestra in figura 2.2, può selezionare il dataset da cui proviene il file o la cartella da analizzare. Se la selezione effettuata non contiene dati nativi di Lifetracker, viene chiamato il metodo *parse()* del rispettivo parser, che effettua concretamente la traduzione dei dati nel formato desiderato (si veda la figura 2.10). A questo punto, i dati possono essere elaborati secondo le modalità descritte nei paragrafi precedenti.

Una volta che il processo di elaborazione è terminato, si giunge nella seconda finestra della GUI, dove sono disponibili grafici riguardanti i dati elaborati, opzioni di visualizzazione di sottoinsiemi dei dati stessi, parametri da impostare il clustering e pulsanti per procedere con la classificazione. In figura 2.9 è possibile vedere un esempio di tale finestra.

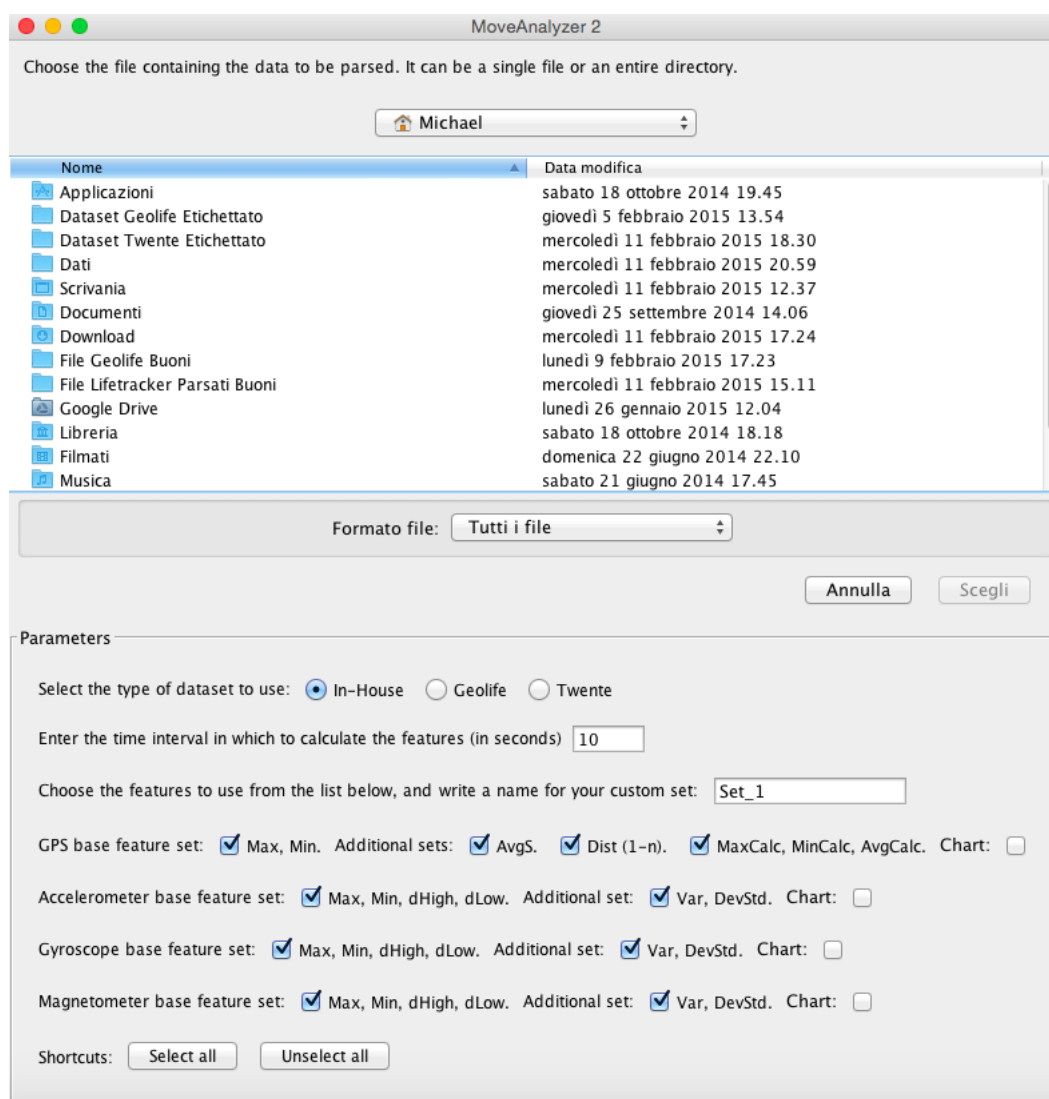


Figura 2.2: Prima finestra dell'interfaccia grafica del software desktop.

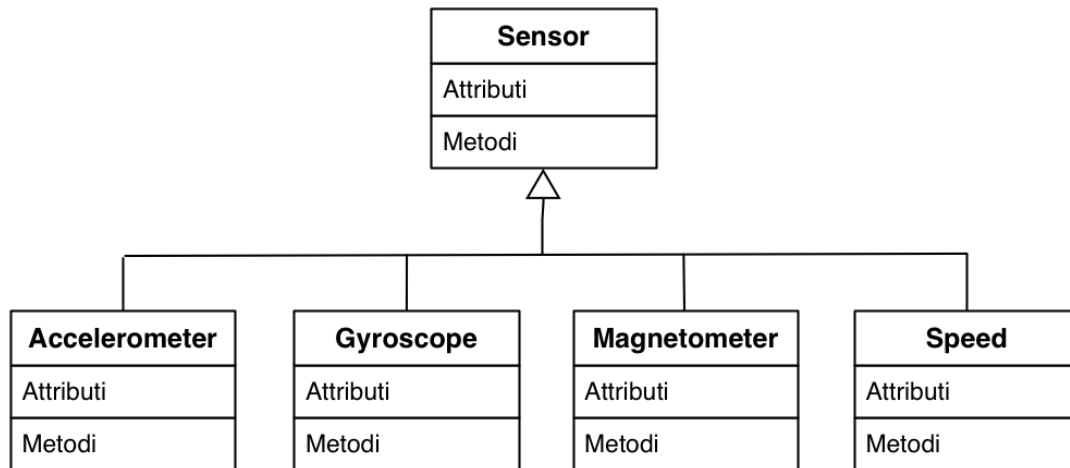


Figura 2.3: Gerarchia delle classi che modellano i quattro sensori.

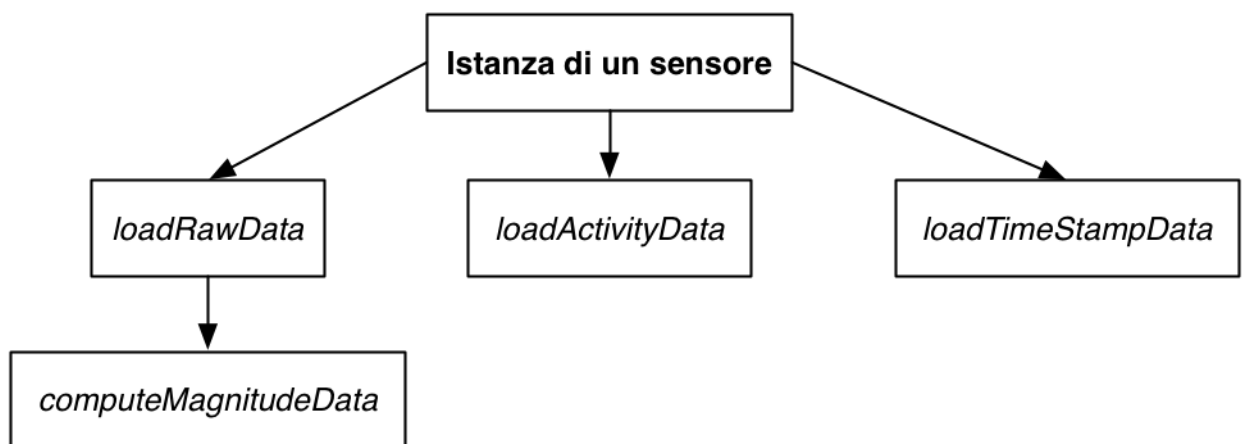


Figura 2.4: Schema dei metodi chiamati dalle istanze dei sensori per elaborare i dati grezzi.

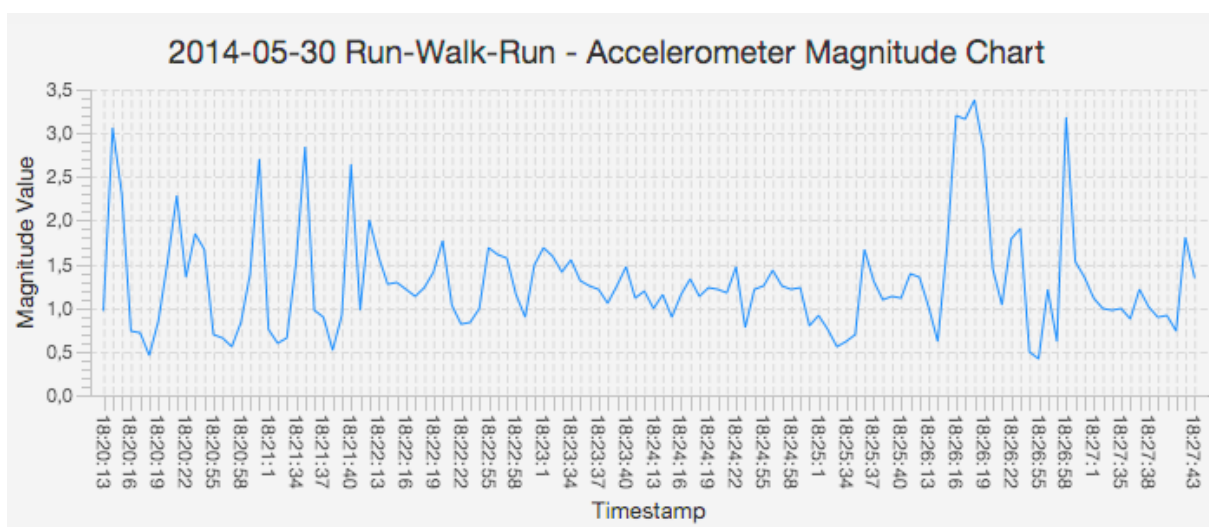


Figura 2.5: Esempio di grafico relativo alla magnitudo dell'accelerometro.

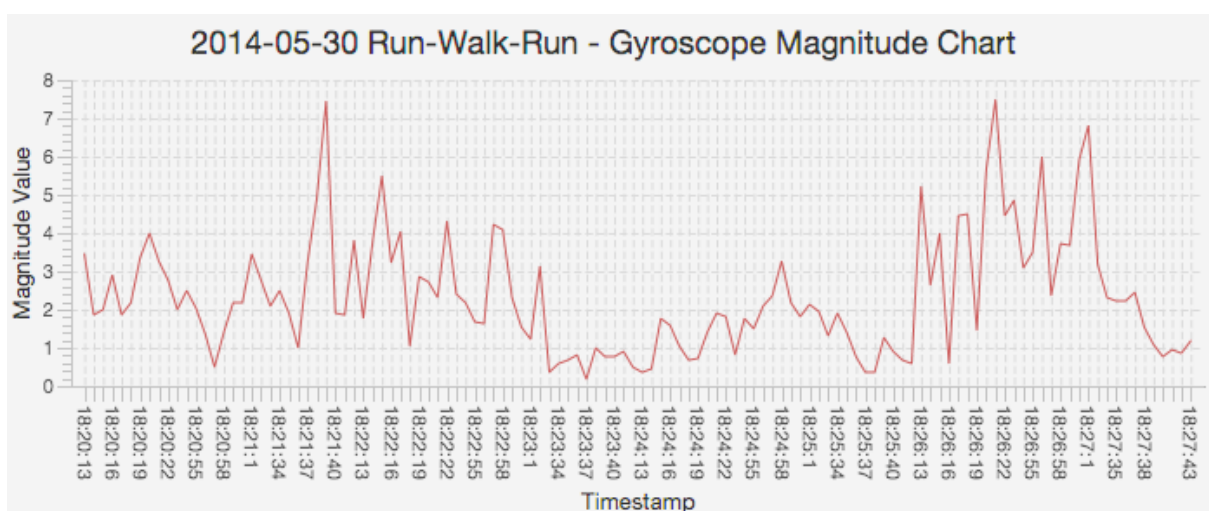


Figura 2.6: Esempio di grafico relativo alla magnitudo del giroscopio.

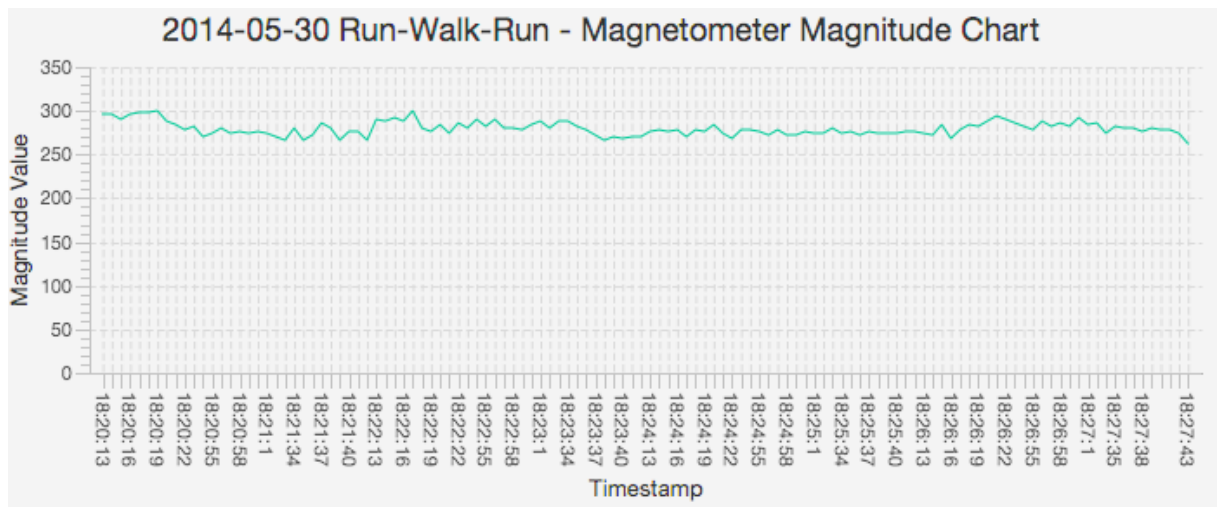


Figura 2.7: Esempio di grafico relativo alla magnitudo del magnetometro.

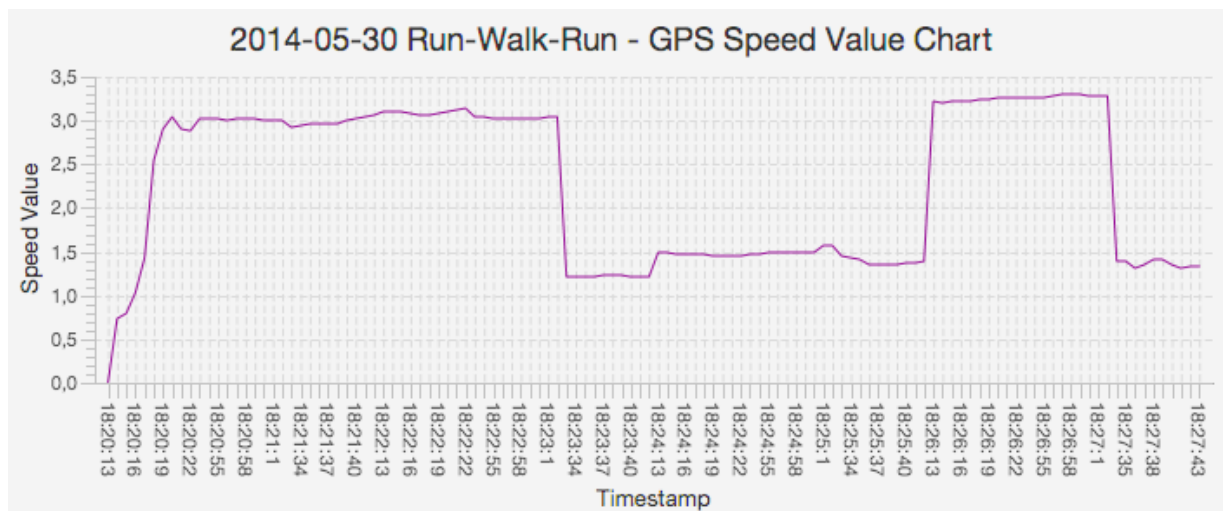


Figura 2.8: Esempio di grafico relativo ai valori della velocità calcolati dal GPS.

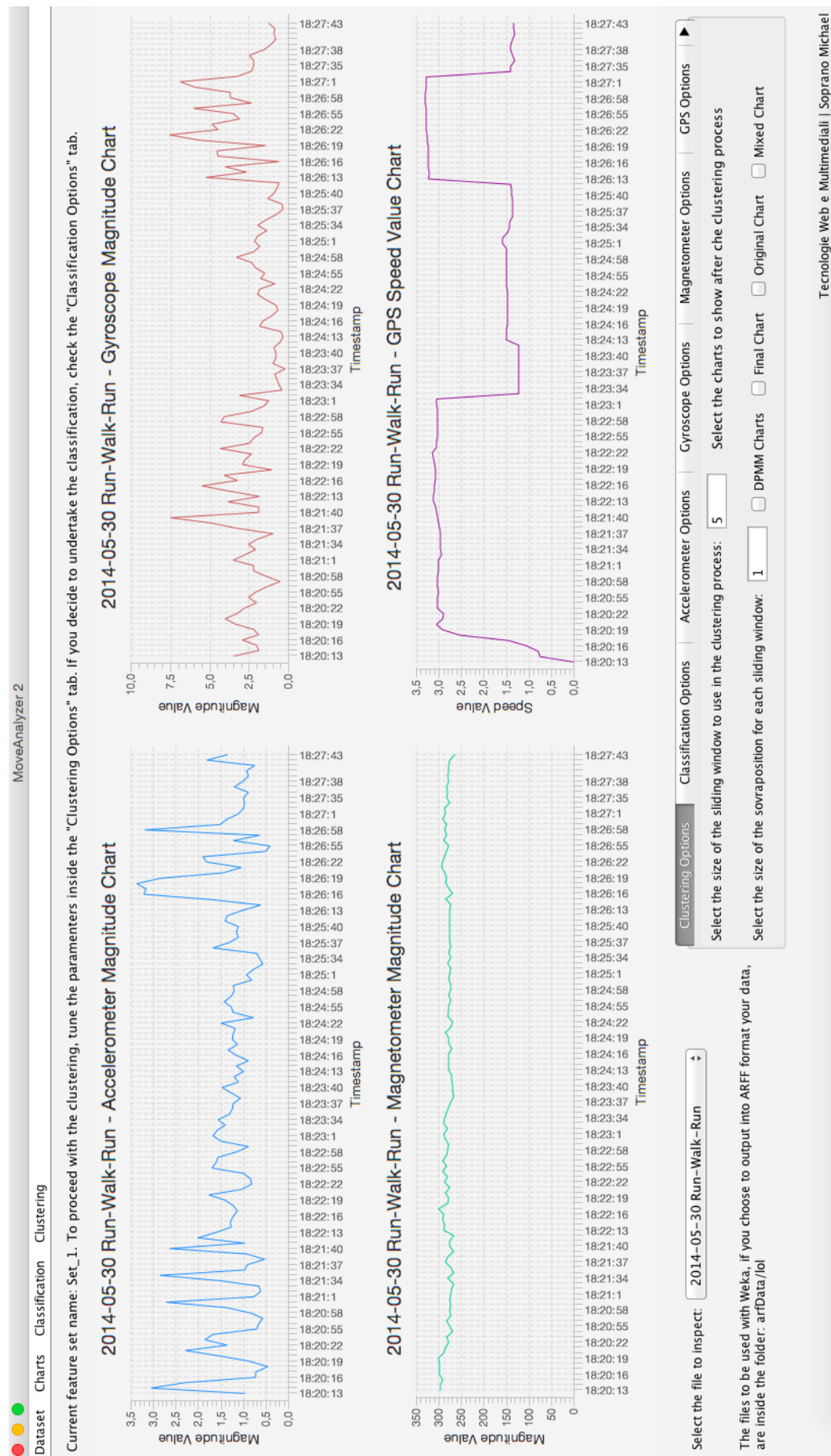


Figura 2.9: Seconda finestra dell'interfaccia grafica del software desktop.

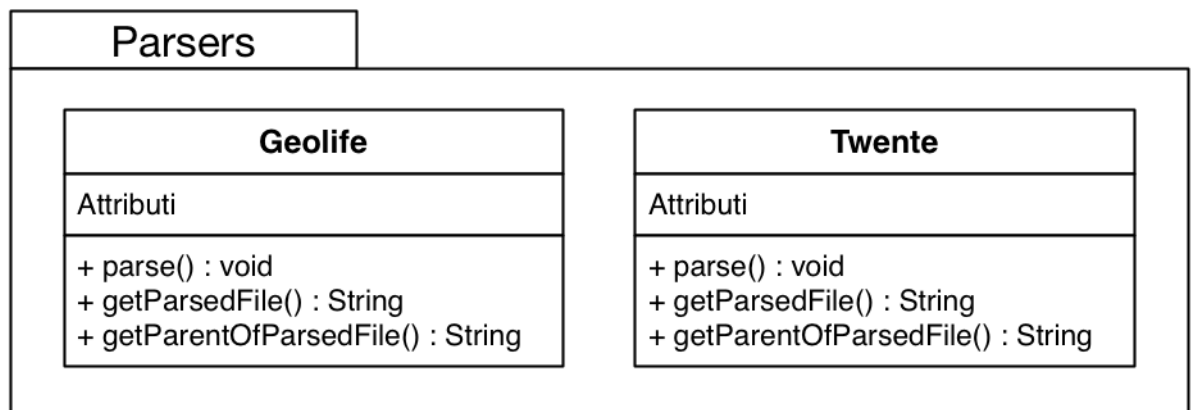


Figura 2.10: Schema del modulo di parsing dei dataset pubblici.

Capitolo 3

Classificazione

Scopo di questo capitolo è descrivere l'analisi dei dati grezzi al fine di riconoscere il mezzo di trasporto utilizzato dagli utenti mediante un approccio supervisionato, la classificazione. Nella sezione 3.1 vengono illustrati i concetti di base necessari per la comprensione della tecnica, mentre nella sezione 3.2 viene descritto il processo di estrazione delle feature dai dati grezzi elaborati. Nella 3.3 viene descritto il software utilizzato per testare i classificatori e il formato richiesto nel quale le feature estratte vanno scritte prima di effettuare la classificazione. Dalla sezione 3.4 alla 3.8 vengono presentate le caratteristiche dei classificatori testati, mentre nella sezione 3.9 viene svolta un'analisi dei risultati ottenuti.

3.1 Concetti base

La classificazione (si veda [1]) è un'attività che consiste nell'assegnare un dato osservato ad una determinata categoria; all'algoritmo di machine learning, il *classificatore*, viene presentato un insieme di istanze classificate a priori (ovvero dati grezzi la cui categoria è nota), il cosiddetto *training set*, che sfrutta per costruire un modello per classificare i nuovi dati che emergeranno, sulla base di quanto appreso dagli esempi; si tratta, dunque, di un approccio di machine learning supervisionato perché il modello, in un certo senso, opera sotto la supervisione fornita dall'informazione contenuta in ciascun esempio del training set. Come si vedrà nelle sezioni seguenti, ci sono diversi modi per costruire tale modello e per selezionare le istanze da raccogliere nel training set.

Ogni istanza è un esempio individuale ed indipendente del concetto da apprendere, ovvero del mezzo di trasporto da categorizzare. Ciascuna istanza viene descritta dai valori di diversi *attributi*, ovvero i vari campi di cui un dato

grezzo è composto. Tali campi possono essere numerici o nominali; il secondo caso si ha quando il valore dell'attributo è tratto da un set finito di possibilità e viene utilizzato come etichetta. Nel dataset utilizzato ogni istanza viene descritta da venti attributi diversi. Una volta costruito il modello a partire dal training set, al classificatore vengono fornite le istanze rimanenti del dataset; il mezzo di trasporto (la categoria in cui ricade il dato grezzo) viene inferita sulla base di quanto modellato.

3.2 Estrazione delle feature

Prima di procedere con la classificazione è necessario un ulteriore processo da applicare ai dati elaborati, l'*estrazione delle feature*. Una feature si può definire come una caratteristica utile da estrarre, o calcolare, rispetto ad un sottoinsieme di dati elaborati del dataset di partenza. Un *feature vector* è un vettore dove ogni elemento è una feature calcolata su tale sottoinsieme dei dati elaborati. Le istanze che vengono fornite in input al classificatore, dunque, sono un insieme di feature vector; ogni attributo di tali istanze sarà una feature calcolata. Il vantaggio di questo processo è che si riduce la dimensione del dataset da classificare, dato che un feature vector riassume più dati grezzi originali, e il numero di attributi da considerare nella classificazione, poiché le feature contenute nel feature vector saranno in numero minore rispetto a quelli originali. Non a caso, l'attività di estrazione delle feature è una forma di riduzione della dimensione, tecnica citata nella sezione 1.3.1.

Nel software l'estrazione viene effettuata chiamando un metodo, contenuto nella classe modellante ciascun sensore; espandendo l'albero delle chiamate in figura 2.4, si ottiene quello descritto in figura 3.1.

Le feature da estrarre sono state decise a priori, in modo euristico. A questo punto diventa chiaro lo scopo dei parametri impostabili dall'utente che si notano nell'interfaccia in figura 2.2.

Un parametro obbligatorio è l'intervallo temporale con cui deve ripetersi il calcolo delle feature, nell'intera sessione di registrazione; confrontando il timestamp dei dati grezzi, ottenuto grazie alla chiamata al metodo *load-TimeStampData*, è possibile capire quali di essi concorrono alla formazione del feature vector per ciascun intervallo; ad un minore intervallo temporale corrisponde un maggior numero di feature vector, e viceversa. Alcune delle checkbox presenti nell'interfaccia consentono di indicare per quali sensori, tra i quattro disponibili, calcolare un set base di feature, mentre le rimanenti consentono di specificare set aggiuntivi da considerare; se la checkbox che indica il sensore non è selezionata, i set aggiuntivi non vengono calcolati. Le varie combinazioni di tali parametri portano ad ottenere diversi set di feature (e

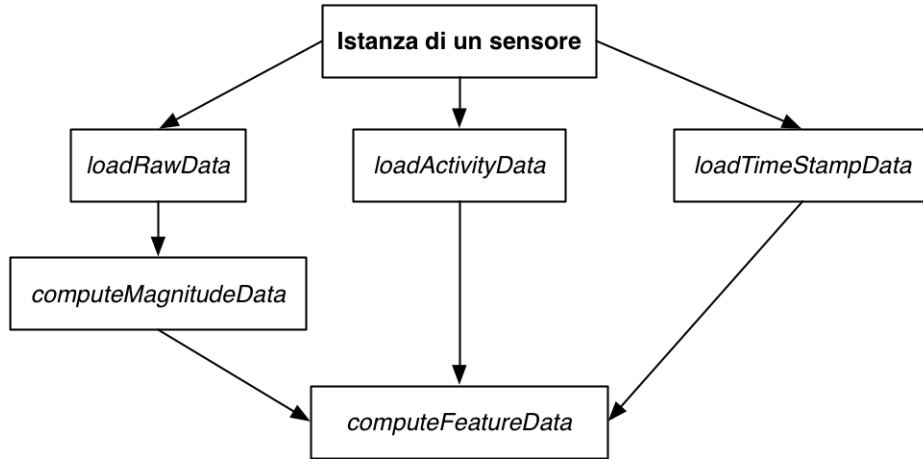


Figura 3.1: Schema delle chiamate effettuate per l'estrazione delle feature.

di relativi feature vector) calcolate; per tale motivo è presente un campo di testo dove poter indicare il nome del set scelto; in tal modo, un utente può produrre tutti i set che desidera, salvarli su file, come illustrato nella sezione 3.3, e testarne le performance nella classificazione.

Le feature calcolate appartenenti ai set base di accelerometro, giroscopio e magnetometro, per ciascuno di essi, sono:

- *Valore massimo*: la magnitudo maggiore tra quelle calcolate per i dati grezzi dell'intervallo corrente;
- *Valore minimo*: la magnitudo minore tra quelle calcolate per i dati grezzi dell'intervallo corrente;
- *Distanza euclidea tra valori oltre due soglie*: una volta ottenuto il valore massimo, si pone una soglia pari al valore massimo stesso meno un quarto del suo valore, sotto la quale i valori di magnitudo sono considerati “alti”; una volta ottenuto il valore minimo, invece, viene posta una seconda soglia, pari al valore minimo stesso con la somma di un quarto del suo valore, oltre la quale i valori di magnitudo vengono considerati “bassi”. Una volta identificati tutti i valori alti e bassi, viene calcolata la distanza tra ciascuna coppia di punti identificati da tali valori (il valore alto n con il valore alto $n+1$, il valore alto $n+1$ con il valore alto $n+2$, ecc.) con la seguente formula:

$$d(a, b) = \sqrt{[(x_2 - x_1)^2] + [(y_2 - y_1)^2]} \quad (3.1)$$

Infine, viene fatta una media di tutte le distanze calcolate. L'idea che sottintende il calcolo di questa feature è che se, per esempio, tale distanza media tra picchi alti o bassi risultasse essere ridotta, l'utente potrebbe essere a fare una corsa a piedi, caratterizzata da brusche variazioni di accelerazione o inclinazione. Viceversa, se tale distanza fosse elevata, l'utente potrebbe trovarsi in automobile, ad una velocità stabile; di conseguenza, non si avrebbero variazioni elevate dei valori grezzi in un ridotto intervallo temporale, e non verrebbero generati picchi alti o bassi. Si tratta, dunque, di una feature in grado di fornire un indizio più sottile per la classificazione dei dati.

Le feature calcolate appartenenti ai set aggiuntivi di accelerometro, giroscopio e magnetometro, per ciascuno di essi, sono:

- *Varianza*: si tratta di una funzione che fornisce una misura della variabilità dei dati rispetto al valor medio. Si calcola con la seguente formula:

$$\sigma^2 = \frac{(X_1 - E(X) + \dots + X_n - E(X))}{n} \quad \text{dove } E(X) \text{ è il valor medio.} \quad (3.2)$$

- *Deviazione Standard*: è un indice di dispersione che consente di esprimere la variabilità dei dati rispetto ad un indice di posizione, che, in questo caso, è sempre il valor medio. La differenza rispetto alla varianza è che quest'ultima ha un'unità di misura quadratica rispetto ai dati, mentre la deviazione standard ha la stessa unità di misura dei dati stessi. Si calcola con la seguente formula:

$$\sigma = \sqrt{\sigma^2} \quad \text{dove } \sigma^2 \text{ è la varianza.} \quad (3.3)$$

Le feature calcolate appartenenti al set base del GPS, invece, sono:

- *Valore massimo*: il maggior valore relativo alla velocità tra quelli estratti appartenente all'intervallo corrente;
- *Valore minimo*: il minor valore relativo alla velocità tra quelli estratti appartenente all'intervallo corrente;
- *Velocità media*: una media dei valori di velocità rilevati appartenenti all'intervallo corrente;

- *Distanza tra primo e ultimo punto GPS*: i dati elaborati per il GPS, come illustrato nella sezione 2.2, comprendono anche latitudine e longitudine; a partire da questi valori è possibile calcolare la distanza in metri tra due punti. Concretamente, viene calcolata per ciascuna coppia di punti dell'intervallo corrente (n con n-1, n-1 con n-2, ecc.) ed infine sommata, per ottenere il valore di essa tra primo e ultimo punto dell'intervallo stesso. Il calcolo viene effettuato sfruttando la formula di Haversine¹; poste latitudine e longitudine come ϕ e λ e sapendo che R corrisponde al raggio terrestre (6,371 km), si ha che:

$$haversine = \sin^2(\Delta\phi/2) + \cos\phi_1 * \cos\phi_2 * \sin^2(\Delta\lambda/2) \quad (3.4)$$

$$c = 2 * \tan^{-1} 2(\sqrt{haversine}, \sqrt{(1 - haversine)}) \quad (3.5)$$

$$d = R * c \quad (3.6)$$

- *Valore massimo e minimo della velocità calcolata dai valori di tempo e distanza*: i rilevamenti della velocità effettuati dal GPS possono essere imprecisi, per cui può essere utile calcolare la velocità a partire dai valori di distanza e tempo, per poi individuare massimo e minimo, al fine di effettuare un confronto con quelli effettivamente rilevati dal sensore. La formula è la seguente:

$$v = d/t \quad (3.7)$$

- *Velocità media calcolata*: una media dei valori di velocità calcolati a partire da quelli di distanza e tempo nell'intervallo corrente.

Nell'ipotesi che tutti i sensori e tutti i set aggiuntivi di feature siano selezionati nell'interfaccia, i feature vector finali saranno così formati:

```
maxAcc,minAcc,dHighAcc,dLowAcc,VarAcc,DevStdAcc,maxGyro,minGyro,
dHighGyro,dLowGyro,VarGyro,DevStdGyro,maxMag,minMag,dHighMag,
dLowMag,VarMag,DevStdMag,maxS,minS,avgS,Dist(1-n),maxCalcS,
minCalcS,avgCalcS
```

Tali vettori, quindi, vengono forniti in input ai vari classificatori implementati all'interno di Weka, descritto nella sezione seguente.

¹Consente di calcolare la distanza tra qualsiasi coppia di punti sul globo terrestre, sfruttando i principi della trigonometria sferica.

3.3 Weka

Per testare le performance delle varie tipologie di classificatori, ci si è affidati ad un software esterno, *Weka*, ampiamente descritto in [1]; esso fornisce l'implementazione di un vasto insieme di algoritmi di machine learning, senza limitarsi ai classificatori, da applicare al proprio dataset; fornisce, inoltre, dei tool di preprocessing dei dati, qualora fosse necessario, e strumenti per visualizzare le caratteristiche dell'algoritmo testato e valutarne le performance e l'output. Si compone di quattro interfacce grafiche distinte, chiamate *Explorer*, *Experimenter*, *Knowledge Flow* e *Sample CLI*. Per tutte le operazioni descritte nel seguito, è stata utilizzata solo la prima di esse.

Le feature estratte, dunque, saranno il dataset fornito in input ai classificatori implementati all'interno di Weka. Prima di procedere, però, è necessario un ulteriore passaggio, che consiste nella scrittura dei feature vector in un formato d'interscambio particolare richiesto da esso; tale formato è chiamato *ARFF*. Il formato ARFF è un modo standard per rappresentare dataset; il file corrispondente si compone di tre parti; la prima è una singola riga dove si indica il nome del dataset con la parola chiave *@relation*. La seconda parte è la dichiarazione degli attributi di cui si compongono le istanze del dataset, mediante la parola chiave *@attribute*; Weka riconosce attributi numerici e nominali; i primi si ottengono aggiungendo la parola chiave *numeric*, mentre i secondi si ottengono indicando le etichette che possono assumere, fra parentesi graffe. La terza parte del file consiste nell'elenco di tutte le istanze, dove il valore di ciascun attributo è separato da una virgola; tale parte comincia con la parola chiave *@data*. I commenti si indicano con il simbolo della percentuale; è possibile, infine, rappresentare valori mancanti inserendo un punto di domanda. Espandendo ancora una volta lo schema in figura 3.1, si ottiene l'albero di chiamate visibile nella 3.2. Il metodo *featureDataToARFF* prende in input i feature vector calcolati; successivamente, scrive le prime due parti del file, indicando così il nome del dataset e tutti gli attributi descriventi i vettori stessi sulla base dei parametri impostati nell'interfaccia; tali parametri sono modificabili per mezzo del menù presente nella finestra di visualizzazione dei grafici dei dati grezzi elaborati e delle opzioni riguardanti essi. Una volta fatto ciò, il metodo scrive la terza parte del file, inserendo ciascun feature vector in una nuova riga del file stesso.

Terminata la sequenza di passaggi precedentemente descritta, si ottiene il file ARFF personalizzato sulla base delle esigenze dell'utente, pronto per l'uso con Weka. Il codice sorgente 3.1 consiste in un file ARFF d'esempio prodotto a partire da un file di dati grezzi presente nel dataset, costituito da cinque minuti di rilevazioni, dove l'intervallo temporale per il calcolo delle feature è pari a dieci secondi; le feature scelte, inoltre, sono quelle base di

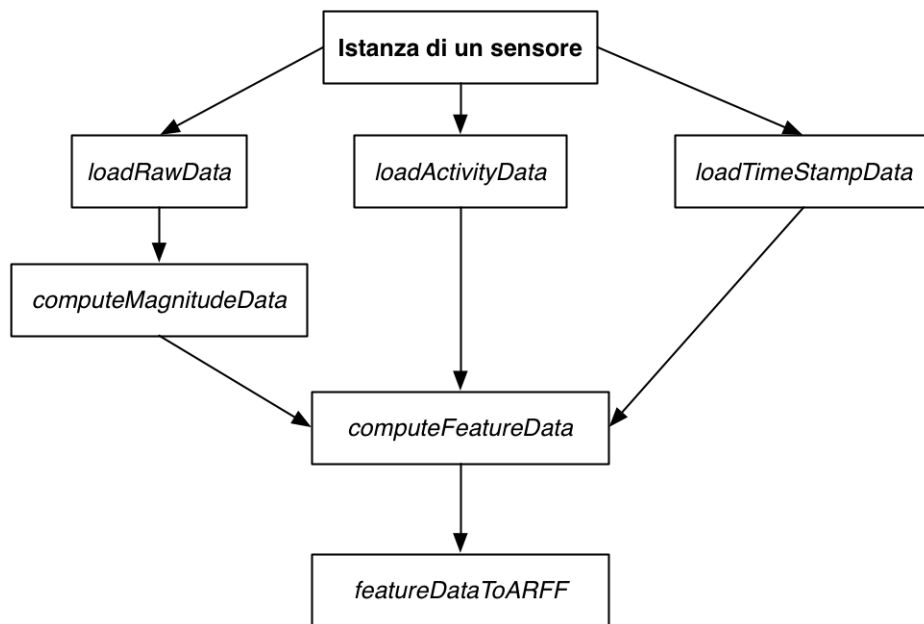


Figura 3.2: Schema delle chiamate effettuate per scrivere le feature estratte in un file ARFF.

GPS e accelerometro. Come si può notare, l'ultimo valore di ogni riga è quello dell'attributo nominale.

@RELATION movement

@ATTRIBUTE maxAcc NUMERIC
 @ATTRIBUTE minAcc NUMERIC
 @ATTRIBUTE dHighAcc NUMERIC
 @ATTRIBUTE dLowAcc NUMERIC
 @ATTRIBUTE maxS NUMERIC
 @ATTRIBUTE minS NUMERIC
 @ATTRIBUTE activity {r,w}

@DATA

3.05578,0.44846,0.62678,0.67851,3.04999,0.0,r
 2.69849,0.557,0.0,1.1765,3.02999,3.0,r
 2.83602,0.51081,2.50182,1.69267,3.05999,2.93,r
 1.76929,0.8141,2.35611,1.0104,3.14,3.06999,r
 1.69553,0.82799,1.17119,1.51289,3.04999,3.01999,r
 1.5546,1.0527,1.13527,0.91605,1.23,1.21,w

```

1.4809,0.89158,1.01047,0.9167,1.49,1.46,w
1.43317,0.75718,0.83886,1.02205,1.57,1.48,w
1.67625,0.5572,1.51575,0.66837,1.46,1.35,w
3.37514,0.60887,0.79127,2.69657,3.26999,3.21,r
3.17527,0.41533,0.0,1.00484,3.29999,3.26999,r
1.80913,0.74411,0.54844,0.88729,1.40999,1.30999,w

```

Sorgente 3.1: File ARFF d'esempio.

Il file ARFF del set di feature corrente si può produrre cliccando sul pulsante apposito nella tab “Classification Options” presente nella finestra in figura 2.9. Una volta prodotti i file, dunque, si può procedere con la classificazione, andando nella tab *Classify* dell'interfaccia *Explorer*. Tuttavia, bisogna fare un'ulteriore considerazione; come spiegato nella sezione 3.1, ai classificatori va fornito un training set da utilizzare per costruire il modello; a Weka, invece, viene fornito il file ARFF contenente tutti i dati; bisogna, dunque, indicare la modalità mediante la quale distinguere, nell'insieme dei dati, il training set da quelli verranno classificati mediante il modello costruito. Tali modalità sono:

- *Use training set*: consiste nell'effettuare la classificazione sull'intero dataset fornito in input, che allo stesso tempo è anche il training set. Come ci si potrebbe aspettare, la percentuale di istanze correttamente classificate è pari al 99-100%; si tratta di uno scenario molto ottimistico sulle future performance del classificatore stesso che soffre pesantemente di *Overfitting*².
- *Supplied test set*: per la costruzione del modello ci si affida ad un determinato training set contenuto su file, assemblato sulla base di determinate scelte. Una scelta del genere potrebbe permettere di ottenere una stima imparziale delle performance.
- *Percentage split*: viene presa la percentuale indicata del dataset fornito per costruire il training set, mentre la percentuale rimanente viene classificata. Solitamente, due terzi del dataset sono una buona percentuale.
- *Cross validation*: Il principio su cui si fonda questa modalità è che ogni classe del dataset debba essere rappresentata nella giusta proporzione nel training set e nel dataset. L'idea generale, dunque, è ripetere il

²Si parla di overfitting quando l'allenamento del classificatore viene fatto con un numero troppo elevato o troppo ridotto di esempi; il modello costruito si adatta, così, a caratteristiche specifiche del training set che non sono presenti nel resto dei dati; il classificatore, dunque, non è in grado di *generalizzare*, ottenendo risultati fuorvianti, poiché è troppo “su misura” rispetto al training set stesso.

processo di costruzione del training set e classificazione del dataset più volte, con campioni differenti. Per far ciò, i dati vengono divisi in n partizioni chiamate *folds*, di dimensione uguale; ogni fold, a turno, è usato per la costruzione del training set, mentre i rimanenti sono usati nella classificazione. Si ripete la procedura per n volte; in tal modo, ogni fold viene usato esattamente una volta per entrambi i compiti. Da qui, il nome di validazione incrociata. In letteratura, dieci viene indicato come un buon numero di partizioni da utilizzare. Tale modalità è quella utilizzata nel testing di tutti i classificatori descritti in questo lavoro.

A questo punto tutti gli elementi necessari per lavorare con Weka e i passaggi preparatori effettuati sui dati elaborati sono stati descritti; nel seguito viene approfondito il funzionamento dei classificatori utilizzati per testare le feature estratte, secondo quanto descritto in [1].

3.4 Classificatori basati sulle regole

La prima categoria di classificatori sviluppata è quella basata sulle *regole di classificazione* derivate da *albero di decisione*. In un albero di decisione, ogni nodo interno consiste in un test di un particolare attributo dei dati di input rispetto ad una soglia; le foglie dell'albero forniscono la classificazione finale a tutti i dati che "raggiungono" la foglia stessa. Se l'attributo testato è numerico, quello che viene eseguito è un test che risponde alla domanda "maggiore di" o "minore di" rispetto alla soglia stessa, ottenendo dunque due possibili ramificazioni dell'albero di decisione. Solitamente, un attributo numerico viene testato più volte in un cammino dalla radice ad una foglia. Se è nominale, invece, viene creato un ramo per ciascuno dei suoi possibili valori e non verrà preso nuovamente in considerazione all'interno di tali rami. Ricapitolando, un'istanza sconosciuta attraversa un determinato cammino sull'albero di decisione e viene testata rispetto a delle soglie costanti³, finché non giunge in una foglia, la quale assegnerà all'istanza stessa una categoria.

Le *regole* sono un'alternativa agli alberi di decisione; sono caratterizzate da un *antecedente*, che consiste in una serie di test, corrispondenti ad un cammino che raggiunge un nodo interno dell'albero di decisione, e da un *conseguente*, che assegna la classe all'istanza testata, in modo analogo alle foglie dell'albero stesso. Solitamente, gli antecedenti vengono applicati secondo un AND logico, per giungere al conseguente.

³Nonostante sia il modo più semplice, va detto che non è l'unico possibile per la costruzione di un albero di decisione. Tuttavia, gli altri modi non vengono approfonditi in questo lavoro

A questo punto è chiaro come un set di regole di classificazione possa essere derivato da un albero di decisione; ogni nodo interno genera una regola, composta da tutti gli antecedenti in and logico appartenenti ai nodi attraversati lungo un cammino dalla radice al nodo interno stesso, mentre le foglie forniscono i conseguenti. A volte può essere svolta un'operazione di *pruning*⁴ sull'albero, poiché spesso risulta essere più complesso del necessario.

3.4.1 OneR

OneR, o meglio, *One Rule*, è un classificatore che si occupa di trovare regole di classificazione da un set di istanze, basandosi un albero di decisione con un solo livello di profondità, come descritto in [1]. Viene preso in considerazione ogni possibile valore presente nelle istanze del training set, per ogni attributo, e ad ognuno di tali valori viene assegnata la classe che occorre il maggior numero di volte nelle istanze del training set.

Ogni attributo, dunque, genera un set differente di regole, una per ogni valore disponibile dell'attributo stesso, dove ogni set corrisponde ad un albero di decisione di profondità uno; nella radice viene posto il valore analizzato e i due figli contengono le due classi che rispondono alle domande “maggiore di” e “minore di” rispetto a quanto contenuto dalla radice stessa, per ogni istanza del training set. Una volta fatto ciò si valuta quante volte ciascuna regola classifica in modo errato le istanze del training set stesso e, infine, si sceglie quella che fornisce le prestazioni migliori; da qui, il nome One Rule; si ottiene quella regola, quel confronto, che genera meno errori per ciascun attributo e infine si sceglie quello migliore (ovvero l'attributo che globalmente sbaglia meno). Per chiarire quanto detto, nel seguito si può osservare uno pseudocodice per l'algoritmo.

Per ogni attributo

```
Per ogni valore dell'attributo, applica la seguente regola:  
  conta quante volte ogni classe compare con quel valore  
  trova la classe più frequente  
  assegna tale classe al valore dell'attributo
```

```
Calcola quante volte le regole applicate su ogni valore sbagliano  
Scegli la regola che sbaglia meno
```

A titolo d'esempio, nel seguito è possibile osservare l'esito di One Rule applicato sul file ARFF descritto nella sezione precedente; l'attributo migliore risulta essere maxS e il valore più frequente (la regola migliore) per la classe m

⁴Consiste nel ridurre la profondità di un albero, ovvero il numero di passi da fare in un cammino dalla radice ad una foglia.

risulta essere -0.5, mentre per la classe r è 2.225 ecc. Sulla base di questo modello, se un'istanza da classificare avesse, come valore di maxS, 41.12, verrebbe assegnata alla classe t.

maxS:

```

< -0.5    -> m
< 0.095684999999999999    -> t
< 2.079985    -> w
< 2.225    -> r
< 2.31743    -> w
< 3.254995    -> r
< 4.539995    -> c
< 6.98048    -> bc
< 39.0106    -> c
>= 39.0106    -> t

```

(736/864 instances correct)

3.5 Classificatori basati su alberi

Vi sono una serie di classificatori che si basano, nuovamente, sul concetto di albero di decisione, dove, a differenza dell'algoritmo One Rule, l'albero viene costruito mediante un approccio *dividi-e-conquista*, a sua volta esteso per gestire problemi quali valori mancanti per uno più attributi, *pruning* per ridurre la profondità dell'albero costruito e quant'altro, che non verranno approfonditi in questo lavoro. In questa tipologia di classificatori, anch'essi descritti in [1], si sposta il focus sul modo in cui costruire l'albero stesso da cui ricavare le regole. Tale albero sarà più complesso di quello dell'approccio descritto nella sezione 3.4.1; per esempio, a meno che le istanze del dataset non siano descritte da un solo attributo, il numero di livelli di profondità sarà maggiore di uno.

L'approccio dividi-e-conquista consiste nell'esprimere la costruzione dell'albero di decisione in modo ricorsivo. La prima cosa da fare è scegliere un attributo da posizionare nella radice dell'albero; successivamente, si genera un ramo per ciascun valore della radice stessa presente nel training set. A questo punto, si prosegue attraversando ciascun ramo e scegliendo, di volta in volta, un nuovo attributo tra quelli non utilizzati per creare un nuovo nodo: a questo punto, si procede ricorsivamente su ognuno di essi. Quando tutte le istanze che giungono in una foglia hanno la stessa classificazione, l'algoritmo smette di scendere in profondità in quel ramo. Il problema consiste nel decidere quale attributo usare per creare un nuovo nodo ad ogni passo della ricorsione.

3.5.1 J48

J48 è l'implementazione di C4.5; viene riassunta in [33]. Si tratta un classificatore basato sugli alberi di decisione costruiti mediante l'approccio dividi-e-conquista. La sua peculiarità consiste nel modo in cui scegliere l'attributo del training set su cui effettuare la suddivisione in rami e la generazione del relativo nodo, ovvero il punto rimasto in sospeso nella sezione precedente. J48, nel fare ciò, sfrutta il concetto di *entropia dell'informazione*; si tratta della quantità media di informazione contenuta in ogni messaggio ricevuto, dove per messaggio si intende un evento, un campione o un carattere estratto da un flusso di dati. L'entropia caratterizza l'incertezza riguardo alla sorgente dell'informazione; la sorgente, solitamente, viene modellata secondo una distribuzione di probabilità. Per creare ciascun nodo dell'albero (dove, come per One Rule, ciascun nodo risponde alla domanda “maggiore di” e “minore di”), l'algoritmo sceglie l'attributo che meglio suddivide gli esempi del training set in sottoinsiemi che vanno ad aumentare il numero di elementi classificati secondo una classe o un'altra. Il criterio secondo cui viene definito qual è il migliore tra essi è l'*information gain* normalizzato, che consiste nella differenza tra l'entropia del training set prima della creazione di un nodo con il nuovo attributo e quella a posteriori; l'attributo con il maggior *information gain* viene considerato il migliore. L'algoritmo procede ricorsivamente nel set di attributi rimanenti. Tale ricorsione, comunque, è regolata da alcuni casi base.

Uno pseudocodice per tale algoritmo è il seguente:

Per ogni attributo *a*

```
    Calcola l'information gain normalizzato suddividendo su a  
    b è l'attributo con l'information gain maggiore  
    crea un nodo dell'albero di decisione su b  
    Ricorri sul set di attributi rimanenti  
    Gli attributi individuati nella ricorsione sono figli di b
```

Nella figura 3.3 è possibile osservare un albero di decisione costruito da J48; per la sessione di classificazione sono state attivate tutte le feature di tutti i sensori, l'intervallo temporale è di dieci secondi e sono stati forniti in input quattro file del dataset. Si noti come maxAcc sia risultato essere l'attributo adatto per la radice dell'albero stesso.

3.5.2 RandomForest

Random Forest è un ulteriore classificatore basato sugli alberi di decisione, ed è un'estensione del concetto sviluppato da J48.

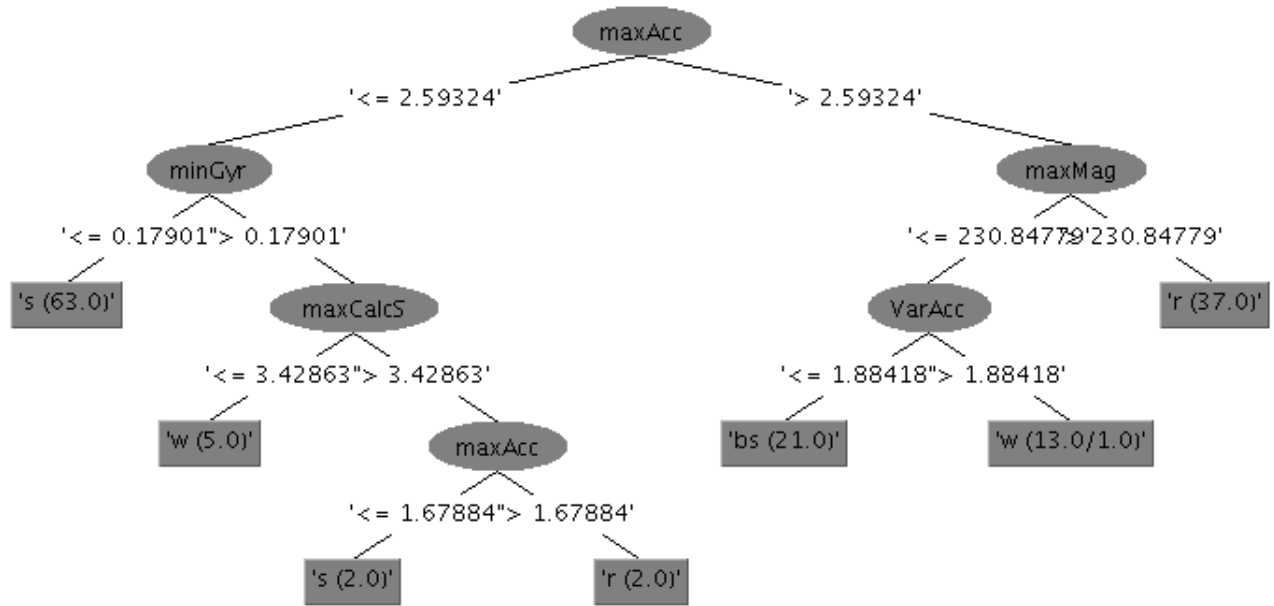


Figura 3.3: Esempio di albero di decisione costruito da J48.

Questo algoritmo, ricevuto in input il training set, genera diversi alberi di decisione contemporaneamente⁵; l'istanza da classificare viene sottoposta a ciascun albero della foresta, e la classificazione finale è la moda⁶ di quelle effettuate da ciascun albero. Il classificatore, dunque, sceglie la classe che ha ricevuto più “voti” all'interno della foresta di alberi di decisione.

Ogni albero della foresta riceve un sottoinsieme casuale delle feature (o attributi) del dataset, e gli esempi utilizzati nella costruzione sono quelli del training set; si può dire, dunque, che ciascun albero classifica in modo diverso e che alla fine vengono tratte delle conclusioni; ogni albero, perciò, può essere visto come uno dei *fold*s, una delle partizioni che genera il risultato finale.

3.6 Classificatori Bayesiani

Un'ulteriore categoria di classificatori è quella basata sul teorema di Bayes e sui concetti della Statistica, descritti nella sezione 1.3.3. Anch'essi hanno ricevuto una trattazione approfondita in [1]. Come si può immaginare, alla base di questi classificatori ci sono le probabilità; a differenza di One Rule, che prende l'attributo migliore e da esso deriva le regole, essi chiamano in causa tutti quelli

⁵Da qui, evidentemente, la foresta presente nel nome dell'algoritmo.

⁶Il valore che compare più frequentemente.

disponibili e in grado di fornire il loro contributo ad un insieme di decisioni; tali decisioni saranno tutte ugualmente importanti ed indipendenti tra di loro. Va precisato che si tratta di una considerazione utopistica, poiché difficilmente i dataset avranno attributi stocasticamente⁷ indipendenti e caratterizzati dalla stessa probabilità; comunque, nonostante tutte le semplificazioni, si ottengono buoni risultati.

3.6.1 NaiveBayes

Le considerazioni illustrate nella sezione precedente hanno portato allo sviluppo di *NaiveBayes*, chiamato così perché assume in modo “naive”, appunto, l’indipendenza degli eventi, “sbloccando” così la possibilità di utilizzare il teorema di Bayes, che richiede tale indipendenza come prerequisito; tale assunzione rende possibile moltiplicare tra di loro le probabilità.

Vi sono, tuttavia, alcuni problemi; per esempio, se il valore di un particolare attributo non occorre almeno una volta per ogni classe, avrà probabilità pari a zero per ciascuna di quelle classi in cui non occorre mai; la moltiplicazione tra le probabilità stesse, dunque, darebbe zero come esito, andando a porre una sorta di “veto” su quell’attributo. Tuttavia, questo problema si può aggirare calcolando la probabilità a partire dalle frequenze cumulate.

Poiché il metodo è semplice, è possibile fare un esempio breve, ma completo, della sua applicazione, applicando pedissequamente il teorema. Supponiamo, quindi, di avere il dataset in figura 3.4.

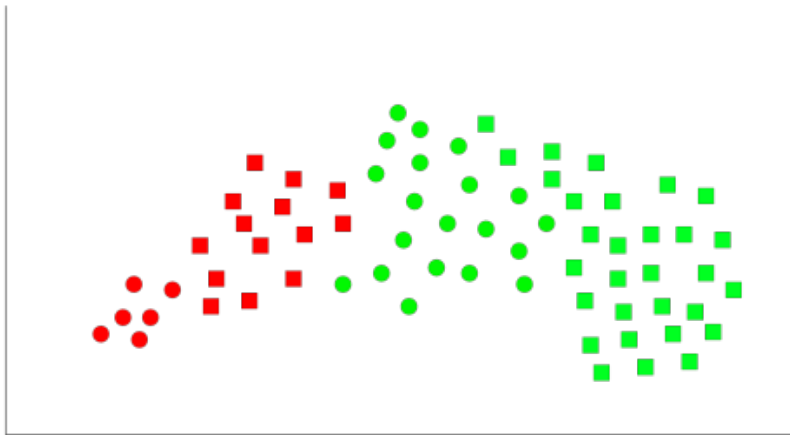


Figura 3.4: Esempio di dataset per Naive Bayes.

⁷Tale termine descrive il fatto che il verificarsi di un evento non modifica la probabilità di un altro di accadere.

Le classi a cui un nuovo oggetto appartenente al dataset può essere assegnato sono, evidentemente, **rosso** e **verde**. Ogni oggetto, inoltre, è caratterizzato dall'attributo forma, che può assumere, come valori, “quadrato” o “cerchio”. Poiché gli oggetti verdi sono in numero maggiore, è ragionevole pensare che il nuovo oggetto abbia una probabilità maggiore di essere **verde**; tuttavia, bisogna considerare anche il contributo dell'attributo, per ottenere buoni valori di probabilità. Proprio qui entra in gioco il teorema di Bayes; si parte dalla probabilità a priori, calcolabile osservando l'esperienza passata (ovvero gli esempi già classificati del training set), per predire l'esito futuro della classificazione sfruttando il teorema, che a sua volta usa a suo vantaggio la probabilità condizionata. Sapendo che gli oggetti in totale sono settanta, si ha che:

$$P(\text{verde}) = \frac{\text{numero di oggetti verdi}}{\text{numero totale di oggetti}} = \frac{50}{70} = 0,71$$

$$P(\text{rosso}) = \frac{\text{numero di oggetti rossi}}{\text{numero totale di oggetti}} = \frac{20}{70} = 0,28$$

Tali valori, dunque, rappresentano le probabilità a priori. A questo punto si possono calcolare le probabilità di osservare ciascun valore dell'attributo forma indipendentemente dalla classe.

$$P(\text{quadrato}) = \frac{\text{numero di oggetti quadrati}}{\text{numero totale di oggetti}} = \frac{44}{70} = 0,62$$

$$P(\text{cerchio}) = \frac{\text{numero di oggetti tondi}}{\text{numero totale di oggetti}} = \frac{26}{70} = 0,37$$

A questo punto si possono ricavare le probabilità condizionate, o a posteriori.

$$P(\text{quadrato}|\text{verde}) = \frac{P(\text{quadrato} \cap \text{verde})}{P(\text{verde})} = \frac{0,42}{0,71} = 0,6$$

$$P(\text{cerchio}|\text{verde}) = \frac{P(\text{cerchio} \cap \text{verde})}{P(\text{verde})} = \frac{0,28}{0,71} = 0,4$$

$$P(\text{quadrato}|\text{rosso}) = \frac{P(\text{quadrato} \cap \text{rosso})}{P(\text{rosso})} = \frac{0,2}{0,28} = 0,7$$

$$P(\text{cerchio}|\text{rosso}) = \frac{P(\text{cerchio} \cap \text{rosso})}{P(\text{rosso})} = \frac{0,085}{0,28} = 0,3$$

Quindi, applicando il teorema di Bayes, si possono ottenere le probabilità che un oggetto venga assegnato a ciascuna classe sulla base dell'attributo forma.

$$P(\text{verde}|\text{quadrato}) = \frac{P(\text{quadrato}|\text{verde}) * P(\text{verde})}{P(\text{quadrato})} = \frac{0,6 * 0,71}{0,62} = 0,69 = 69\%$$

$$P(\text{rosso}|\text{quadrato}) = \frac{P(\text{quadrato}|\text{rosso}) * P(\text{rosso})}{P(\text{quadrato})} = \frac{0,7 * 0,28}{0,62} = 0,31 = 31\%$$

$$P(\text{verde}|\text{cerchio}) = \frac{P(\text{cerchio}|\text{verde}) * P(\text{verde})}{P(\text{cerchio})} = \frac{0,4 * 0,71}{0,37} = 0,77 = 77\%$$

$$P(\text{rosso}|\text{cerchio}) = \frac{P(\text{cerchio}|\text{rosso}) * P(\text{rosso})}{P(\text{cerchio})} = \frac{0,3 * 0,28}{0,37} = 0,23 = 23\%$$

Una volta applicato il teorema stesso sul training set dell'esempio, dunque, si può vedere come il modello costruito dal classificatore assegni un nuovo oggetto alla classe **rosso** con il 31% di probabilità ed alla classe **verde** con il 69% se il valore dell'attributo forma è pari a quadrato. Se tale valore, invece, è pari a cerchio, le due percentuali sono del 23% e del 77%.

In questo esempio di base i dati posso essere contati manualmente; in un contesto reale, NaiveBayes modella le istanze (e quindi i valori dei singoli attributi) secondo distribuzioni Gaussiane, di Bernoulli o multinomiali. Va scelta la versione di NaiveBayes che più si adatta alle caratteristiche degli attributi del proprio dataset.

Come si è detto precedentemente, nonostante non tutti i dataset possano andar bene per questo classificatore per via delle dipendenze tra attributi che ne riducono le performance, rimane un approccio semplice che andrebbe sempre testato prima di approfondire schemi di machine learning più sofisticati.

3.6.2 BayesNet

Un classificatore più sofisticato di quello della sezione precedente che, tuttavia, è fondato sulle stesse basi statistiche è *BayesNet*. Si vedano i riferimenti [34] e [35].

Una rete Bayesiana consiste in un modello probabilistico tramite il quale viene rappresentato un insieme di variabili casuali assieme ad eventuali dipendenze esistenti tra di esse mediante un DAG⁸. Già a questo punto si può notare una differenza rispetto al NaiveBayes, dato che quest'ultimo non consente di tener conto di tali dipendenze. Nel contesto della classificazione, l'insieme di variabili casuali consiste nelle feature che caratterizzano le istanze del dataset.

⁸Si tratta di un grafo diretto (o orientato) caratterizzato dall'assenza di cicli. L'acronimo, infatti, sta per direct acyclic graph.

A titolo d'esempio, in figura 3.5 è possibile osservare la rete costruita dal classificatore (anch'esso, come gli altri, implementato in Weka) per alcune delle feature calcolate.

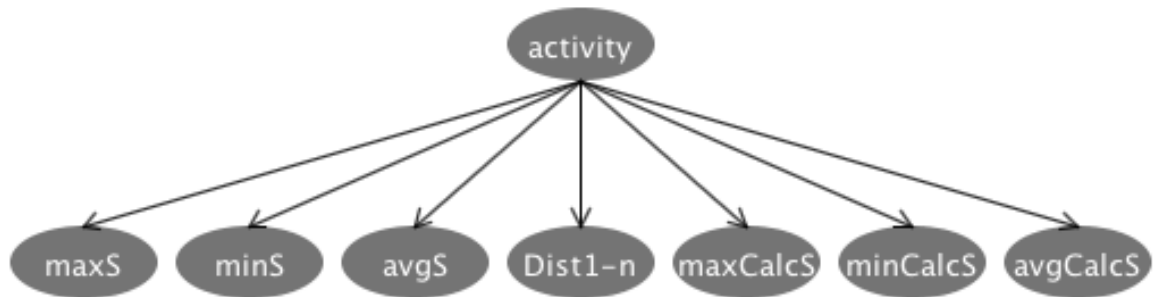


Figura 3.5: Esempio di DAG costruito da BayesNet.

Ogni nodo del grafo, dunque, corrisponde ad una variabile casuale, ovvero ad una feature, mentre gli archi sono le dipendenze. Come si può intuire, solamente l'attributo nominale *activity* dipende dai valori degli altri; tra i restanti attributi, invece, non sussiste alcuna relazione; l'obiettivo della classificazione è proprio scoprire cosa assegnare a tale attributo sulla base degli altri.

Ogni nodo, inoltre, è caratterizzato da una distribuzione di probabilità, che prende in input un particolare insieme di valori del nodo padre (ovvero, i mezzi di trasporto utilizzati nel training set) e restituisce come output il valore di probabilità (o un'ulteriore distribuzione) che caratterizza la variabile presente nel nodo rispetto a ciascun valore del padre stesso. Di volta in volta, quindi, le distribuzioni ottenute verranno utilizzate per classificare le istanze del dataset.

Uno dei modi più semplici per costruire una rete Bayesiana è farlo “manualmente”, basandosi sulla conoscenza del dominio e consiste di tre passaggi:

- Determinare numero e significato delle variabili presenti nel dominio;
- Determinare le dipendenze tra le variabili;
- Determinare i valori delle probabilità condizionate per ciascuna variabile.

Nei casi più complessi diventa estremamente difficile procedere in tal senso, quindi sono desiderabili algoritmi in grado di costruire autonomamente la struttura della rete. Tale problema, però, è NP-completo, quindi sono state sviluppate tutta una serie di euristiche che si dividono in due macro categorie di approcci. Tuttavia, tali approcci non vengono approfonditi in questo lavoro.

3.7 Reti neurali

Una delle tipologie più avanzate di classificatori consiste nelle *Reti Neurali*; per quanto riguarda i riferimenti, si vedano [1], [36] e [37].

Esseri umani ed animali sono in grado di eseguire compiti di classificazione anche molto complessi e ciò viene svolto, nel cervello, a livello neuronale. Tuttavia, la “potenza computazionale” di un singolo neurone non permette ad esso di svolgere in autonomia un compito del genere. Il segreto sta nel fatto che tali neuroni sono estremamente interconnessi tra di loro, ed ogni problema complesso può essere scomposto in sottoproblemi risolvibili dal singolo neurone. Questa idea ha portato allo sviluppo delle reti neurali.

Il singolo neurone della rete viene modellato dal *Perceptron*, o Percettrone. Esso consiste in un algoritmo che consente di sviluppare un classificatore binario, uno strumento che consente di affermare se un’istanza è un buon esempio o meno, senza fornire ulteriori informazioni. In figura 3.6 è possibile vedere la sua definizione.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Figura 3.6: Definizione di un Perceptron.

L’input del Perceptron è x , ovvero un feature vector da classificare. Il Percettrone stesso mappa il vettore in un output binario; w è un vettore di numeri reali utilizzati come pesi, mentre $w \cdot x$ consiste nel prodotto scalare tra i vettori stessi. Il termine b è il cosiddetto bias, un valore costante che non dipende dall’input. Una rete neurale, dunque, è costituita da un insieme di Percettroni interconnessi tra di loro.

on

Ci sono diversi tipi di reti neurali ma nella maggior parte dei casi si fa riferimento al *MultilayerPerceptron*. In questa tipologia di rete i percettroni sono organizzati in una struttura particolare, formata da tre livelli, come si può vedere in figura 3.7. Ciascun livello è un grafo di Percettroni ed ognuno di essi è fortemente connesso⁹ con il successivo; in questo caso, gli archi rappresentano le sinapsi. Tale caratteristica implica che l’output di ogni Percettrone di un determinato livello viene distribuito a tutti quelli del livello successivo; tale

⁹Un grafo è fortemente connesso se esiste un cammino tra ciascuna coppia di nodi del grafo stesso.

rete neurale, inoltre, è di tipo *feed-forward*; ciò vuol dire gli output si muovono solamente in avanti, verso i livelli successivi, senza mai tornare indietro.

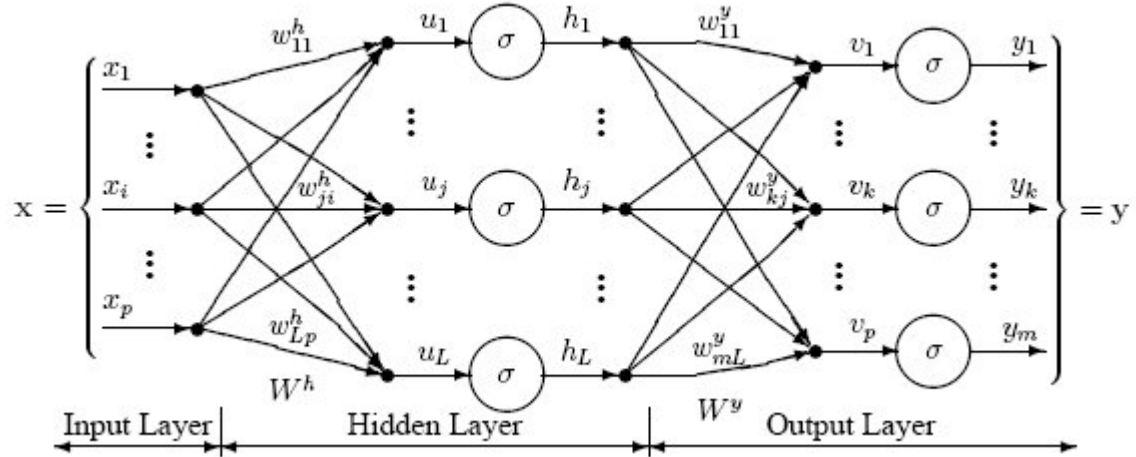


Figura 3.7: Struttura del MultilayerPerceptron.

La rete, dunque, si compone di tre livelli, che ne regolano il funzionamento:

- *Livello di input*: a tale livello viene presentato un feature vector; ci sono N percettroni per N feature numeriche, $N-1$ se una di esse è di tipo nominale. Ogni feature del feature vector stesso viene standardizzata; in tal modo, il valore di ognuna di esse risulta compreso tra -1 e 1 . A questo punto, il livello di input distribuisce ogni risultato a ciascun nodo del secondo livello, quello nascosto, prendendo in considerazione anche il bias. Il bias viene moltiplicato per un peso e sommato al valore che giunge nei vari percettroni nascosti.
- *Livello nascosto*: i valori che giungono dai percettroni di input in ciascun nodo del livello nascosto vengono moltiplicati per un peso (w_{ji}) e sommati assieme, producendo il valore (u_j). Esso viene fornito in input ad una funzione di trasferimento σ , che restituisce il valore (h_j). Ogni h_j viene distribuito a ciascun percettrone del livello di output.
- *Livello di output*: ogni valore che giunge dal livello nascosto viene moltiplicato per un peso (w_{kj}) ed i risultati sono sommati tra di loro producendo un ulteriore valore (u_j). Ogni u_j viene fornito in input ad una funzione di trasferimento σ , come nel livello nascosto, ed i risultati y_j sono l'output della rete neurale. I nodi del livello di output sono il risultato della classificazione.

Le funzioni di trasferimento si occupano della trasformazione del risultato della funzione di somma in un altro formato; nell'ultimo stadio, in particolare, sono il meccanismo che fornisce l'esito della classificazione.

Come si può vedere, la struttura della rete dipende dal dataset da analizzare, mentre i vari pesi menzionati nella descrizione dei tre livelli derivano dall'applicazione dell'algoritmo di *Backpropagation*, descritto dettagliatamente in [1].

3.8 Macchine a vettori di supporto

L'ultima categoria di classificatori testati è quella più avanzata e recente tra quelle descritte in precedenza. Essa è costituita dagli algoritmi chiamati *macchine a vettori di supporto*, o SVM¹⁰; tali classificatori sono nati come alternativa alle reti neurali, e hanno questo nome perché si basano, appunto, sul concetto di *vettore di supporto*. I riferimenti utilizzati per lo studio di tale categoria sono [38] e [1].

Gli algoritmi di tipo SVM, per eseguire la classificazione, selezionano un certo numero di istanze critiche che stanno ai confini di ciascuna classe, ovvero i cosiddetti vettori di supporto, per poi costruire una funzione in grado di separarle nel modo più ampio possibile. Un *modello* SVM consiste nella rappresentazione degli esempi del training set come punti nello spazio, dove ciascun punto viene separato dagli altri dalla funzione trovata, in base alla sua classe. Le nuove istanze da classificare vengono mappate nello spazio descritto dal modello e la classe finale viene predetta in base al lato della suddivisione in cui tali istanze ricadono. In figura 3.8, per esempio, H3 è la funzione che separa le istanze nello spazio nel modo più ampio possibile.

Un possibile problema, sta nel fatto che, in contesti reali d'uso, le istanze dei dataset difficilmente sono linearmente separabili¹¹. Gli algoritmi SVM rimediano a ciò sfruttando funzioni anche non lineari in grado di descrivere tale separazione. L'idea chiave consiste nel trasformare lo spazio delle istanze in nuovo tipo di spazio aggiungendo dimensioni (passare ad uno spazio vettoriale di dimensione maggiore), in modo da ottenere una mappatura delle istanze stesse non lineare. Per via di tale motivo una linea dritta in grado di separare

¹⁰Support Vector Machines.

¹¹Si tratta di una proprietà geometrica esistente tra le coppie di punti di un insieme, nella geometria Euclidea. Supponiamo di avere punti colorati di rosso e punti colorati di blu. Tali punti si dicono linearmente separabili se esiste almeno una linea nel piano dove tutti i punti blu sono un lato di essa e tutti quelli rossi sull'altro. Tale linea, evidentemente, sarà descritta da una funzione.

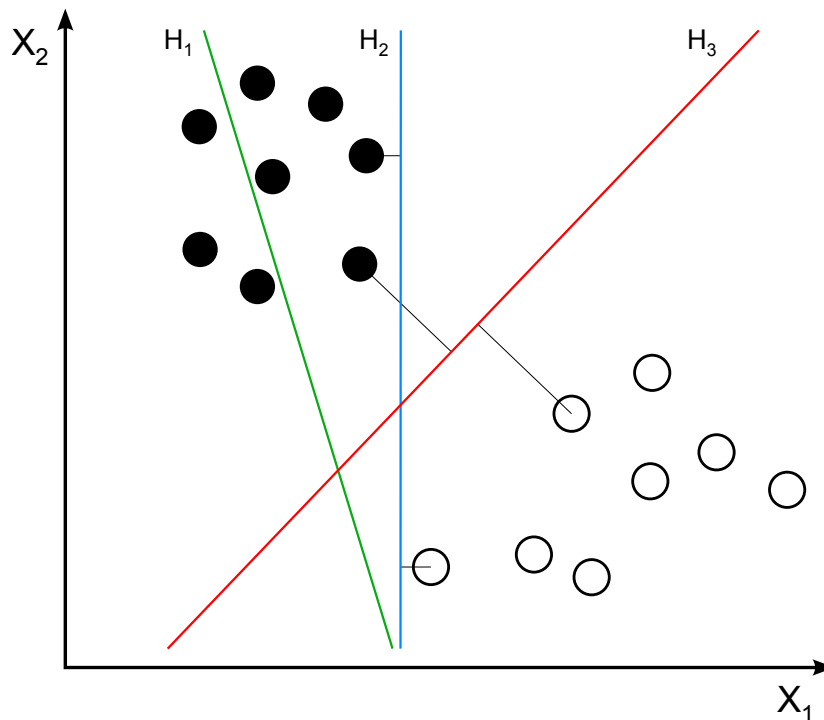


Figura 3.8: Esempio di punti linearmente separabili.

le istanze in modo lineare nel nuovo spazio, può risultare di diverso tipo in quello originale.

Sfruttando i vettori di supporto, dunque, si possono ottenere classificatori lineari e non; Weka, in particolare, mette a disposizione una classe wrapper LibSVM, per integrare al proprio interno una libreria esterna sviluppata da altri programmatori che implementa una tipologia di classificatore sfruttante le SVM stesse.

3.8.1 Maximum-Margin Hyperplane

Nel seguito si intende fornire un rapido esempio di classificatore basato sul concetto di SVM, chiamato *Maximum-Margin Hyperplane*. Va precisato che non è stato testato questo particolare algoritmo nella classificazione delle istanze del dataset di questo lavoro, a differenza di quelli descritti in precedenza; serve, appunto, solamente a titolo esemplificativo per poter dare una maggiore consistenza all'esempio descritto nella sezione precedente.

Si immagini, dunque, di avere un dataset costituito da due classi diverse le cui istanze sono linearmente separabili. Una volta collocate nello spazio,

è possibile trovare un iperpiano¹² che le classifica correttamente; l'obiettivo, però, non è trovarne uno qualsiasi, ma quello di margine massimo; esso fornisce la più grande separazione tra le classi del dataset, motivo per cui è desiderabile trovarlo.

Supponiamo, quindi, di trovare per ciascuna classe del dataset, il poligono convesso che racchiude nel modo più su misura possibile tutte le istanze di essa connettendo tra di loro i punti nello spazio; dato che le due classi sono linearmente separabili, tali poligoni non si sovrapporranno mai. l'iperpiano di margine massimo sarà quello il più lontano possibile dai due poligoni individuati, e le istanze di ciascuna classe che sono più vicine ad esso sono i vettori di supporto. Il set di vettori di supporto, dunque, fornisce proprio l'iperpiano di margine massimo per eseguire la classificazione. Uno schema esemplificativo di tale processo si può osservare in figura 3.9.

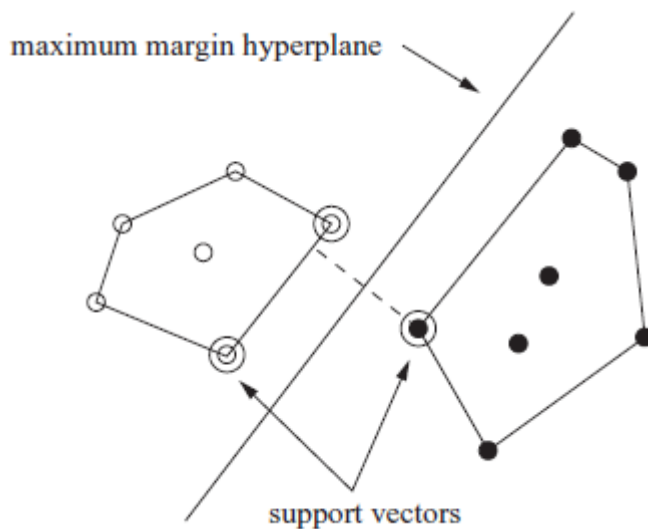


Figura 3.9: Esempio di Maximum-Margin Hyperplane - (Fonte: [1])

3.9 Analisi risultati

In questa sezione si intende presentare gli esiti dei test eseguiti mediante il software Weka per ciascuna tipologia di classificatore descritta nelle sezioni precedenti, con diversi set di feature. Si è scelto di testare queste tecniche

¹²In uno spazio tridimensionale euclideo un piano è un insieme di punti che soddisfa un'equazione lineare e separa i punti rimanenti dell'intero spazio in due semispazi. Analogamente, in uno spazio a quattro o più dimensioni, si può trovare un sottospazio di dimensione $n-1$, chiamato proprio iperpiano, che separa i punti in due insiemi.

di machine learning poiché fanno parte della storia di tale settore. Ciascun set è stato prodotto sfruttando i parametri di produzione impostabili nell'interfaccia grafica del software desktop descritta nel capitolo 2. I diversi set sono stati prodotti come indicato nella tabella 3.1 e testati su una settimana di dati registrati dall'applicazione Lifetracker; tale settimana e tutti gli altri dataset analizzati sono descritti nella tabella 3.2. Il risultato di tale test si può osservare nella tabella 3.3; i valori contenuti in essa, dunque, indicano la percentuale di istanze correttamente classificate per ciascuna tipologia di classificatore testata; in questo modo, viene individuato quello migliore. A questo punto, tutti i dataset rimanenti vengono classificati mediante tale classificatore, ed i risultati sono disponibili nella tabella 3.4. Per avere una visualizzazione migliore del numero di occorrenze per ciascuna classe di mezzo di trasporto presente all'interno del dataset stesso, si può osservare il grafico in figura 3.10, ottenibile mediante un pulsante presente nell'interfaccia grafica del software desktop. La modalità di test impiegata è la cross-validation (si veda la sezione 3.3). Dopo alcuni tentativi, si è notato come l'intervallo temporale più adatto con il quale produrre ciascun feature vettore al fine di discriminare i vari mezzi di trasporto, sia di dieci secondi; perciò, in tutti i set prodotti, ciascun vettore copre tale intervallo temporale. Infine, ciascun classificatore richiede la specifica di parametri per il proprio funzionamento; si è deciso, dunque, di affidarsi ai valori di default impostati in Weka, per evitare di ottenere risultati errati dovuti ad un cattivo uso dei parametri stessi.

In questo paragrafo si intende illustrare alcune considerazioni legate alle tabelle 3.1 e 3.3. In generale, cambiare feature set, come si può notare, influenza le prestazioni dei classificatori in positivo oppure in negativo, siano essi quelli migliori o quelli peggiori. In particolare, se si analizzano i risultati ottenuti con i feature set dal numero uno al numero quattro, i quali consistono in combinazioni di feature del solo GPS, si scopre che il migliore tra essi è il terzo; tale set corrisponde a tutte le feature del GPS stesso meno le velocità calcolate a partire dai valori di latitudine e longitudine. Il quarto set di feature, invece, riduce le prestazioni dei classificatori a causa, probabilmente, dei sei diversi valori di velocità presenti; potrebbe essere sufficiente prendere solo tre di essi, in particolare quelli relativi alla velocità calcolata; questo perché è noto che i valori rilevati direttamente dal GPS sono soggetti ad errori anche piuttosto pronunciati. Per quanto riguarda il feature set cinque, si può notare un buon aumento nelle performance in tutti i classificatori, segno che l'aggiunta dell'accelerometro è una scelta efficace; l'unico a non trarre alcun beneficio è OneRule; il motivo va ricercato nell'estrema semplicità di questa tecnica, che non riesce a sfruttare le nuove informazioni disponibili in nessun caso a causa del suo funzionamento, spiegato nella sezione 3.4.1. Nel sesto feature set ci

sono alcuni miglioramenti ed alcuni peggioramenti; in questo caso può essere utile creare ulteriori feature set spezzando quelli che hanno più di due feature, come per esempio quelle dell'accelerometro, creando gruppi da due; in tal modo l'analisi viene svolta con una granularità più fine portando, probabilmente, ad un miglioramento nei risultati. Ad ogni modo, i classificatori migliori, J48 e RandomForest, rientrano tra quelli che vedono aumentare le loro performance. Il settimo feature set vede l'aggiunta del giroscopio; anche tale operazione si rivela utile, poiché le prestazioni di tutti i classificatori aumentano, circa, dell'uno per cento. I risultati vengono nuovamente migliorati anche con il feature set otto, che comporta l'aggiunta del magnetometro; in questo caso vi sono aumenti per il tre-quattro per cento circa, segno che tale sensore è molto più utile di quanto si pensasse inizialmente. Con i feature set sei, nove e dieci si analizzano, rispettivamente, accelerometro, giroscopio e magnetometro singolarmente, senza l'apporto del GPS; si può notare come il secondo sia il peggiore dei tre, mentre il terzo ottiene un punto percentuale in più rispetto al primo, ad ulteriore riprova della validità del magnetometro. Nei feature set che vanno dal numero undici al numero quattordici vengono analizzate combinazioni degli stessi sensori dei casi precedenti, nuovamente senza GPS; per la maggior parte dei classificatori vi sono le performance globalmente più basse. Con il quindicesimo set vengono analizzati tutti e tre i sensori assieme; sorprendentemente, le prestazioni sono migliori rispetto al quarto feature set, contenente le feature del solo GPS. Si tratta di un fatto molto importante, poiché in una futura applicazione mobile si può consentire all'utente di spegnere lo stesso GPS e sfruttare solamente i tre sensori legati al movimento, ottenendo, per esempio, un cospicuo risparmio di energia della batteria del dispositivo mobile utilizzato senza incidere considerevolmente sulla qualità del risultato finale. Si tratta, dunque, di un fatto che può portare ad utili miglioramenti da un punto di vista più applicativo e meno teorico.

Per concludere, i classificatori migliori per eseguire il riconoscimento del mezzo di trasporto eseguito sui dati forniti dall'applicazione Lifetracker risultano essere RandomForest e J48, con tutte le feature di ciascun sensore ricevute in input. Una buona alternativa, infine, è MultilayerPerceptron, distaccato di tre punti percentuali circa. A questo punto può essere utile una considerazione riguardante l'aspetto implementativo; l'algoritmo di J48, basato sulla costruzione di un albero di decisione, è il più semplice tra i tre, quindi potrebbe essere accettabile la perdita di punti percentuali di istanze correttamente classificate ai fini di ottenere un'implementazione più semplice da realizzare e più efficiente. Le percentuali medie di successo per J48 e RandomForest, per le diverse tipologie di dataset testati, sono:

- *J48 - Geolife*: 79,83%.

- *RandomForest* - *Geolife*: 76,62%.
- *J48* - *Twente*: 56,64%.
- *RandomForest* - *Twente*: 52,31%.
- *J48* - *Lifetracker*: 92,35%
- *RandomForest* - *Lifetracker*: 92,76%

Set	GPS	Accelerometer	Gyroscope	Magnetometer
1	Max Speed, Min Speed	-	-	-
2	Max Speed, Min Speed, Avg Speed	-	-	-
3	Max Speed, Min Speed, Avg Speed, Dist 1-n	-	-	-
4	Max Speed, Min Speed, Avg Speed, Dist 1-n, Max SpeedCalc, Min SpeedCalc, Avg SpeedCalc	-	-	-
5	Max Speed, Min Speed, Avg Speed, Dist 1-n, Max SpeedCalc, Min SpeedCalc, Avg SpeedCalc	Max, Min, dHigh, dLow	-	-
6	Max Speed, Min Speed, Avg Speed, Dist 1-n, Max SpeedCalc, Min SpeedCalc, Avg SpeedCalc	Max, Min, dHigh, dLow, Var, DevSt	-	-
7	Max Speed, Min Speed, Avg Speed, Dist 1-n, Max SpeedCalc, Min SpeedCalc, Avg SpeedCalc	Max, Min, dHigh, dLow, Var, DevSt	Max, Min, dHigh, dLow, Var, DevSt	-

8	Max Speed, Min Speed, Avg Speed, Dist 1-n, Max SpeedCalc, Min SpeedCalc, Avg SpeedCalc	Max, Min, dHigh, dLow, Var, DevSt	Max, Min, dHigh, dLow, Var, DevSt	Max, Min, dHigh, dLow, Var, DevSt
9	Max Speed, Min Speed, Avg Speed, Dist 1-n, Max SpeedCalc, Min SpeedCalc, Avg SpeedCalc	-	Max, Min, dHigh, dLow, Var, DevSt	-
10	Max Speed, Min Speed, Avg Speed, Dist 1-n, Max SpeedCalc, Min SpeedCalc, Avg SpeedCalc	-	-	Max, Min, dHigh, dLow, Var, DevSt
11	-	Max, Min, dHigh, dLow, Var, DevSt	-	-
12	-	-	Max, Min, dHigh, dLow, Var, DevSt	-
13	-	-	-	Max, Min, dHigh, dLow, Var, DevSt
14	-	Max, Min, dHigh, dLow, Var, DevSt	Max, Min, dHigh, dLow, Var, DevSt	-
15	-	Max, Min, dHigh, dLow, Var, DevSt	-	Max, Min, dHigh, dLow, Var, DevSt

Tabella 3.1: Elenco dei feature set prodotti al fine di individuare i classificatori migliori mediante Weka.

Dataset	Sorgenti	Sensori	Giorni	Utenti	Mezzi	Descrizione
GL1	Geolife	GPS	7	1	6	Bus, Subway, Airplane, Taxi, Railway, Walk.
GL2	Geolife	GPS	7	1	2	Subway, Walk.
GL3	Geolife	GPS	7	1	4	Bus, Walk, Car, Taxi.
TW1	Twente	Accelerometer	4	3	6	Walk, Sit, Run, Downstair, Stand, Upstair (Pocket).
TW2	Twente	Gyroscope	4	3	6	Walk, Sit, Run, Downstair, Stand, Upstair (Pocket).
TW3	Twente	Magnetometer	4	3	6	Walk, Sit, Run, Downstair, Stand, Upstair (Pocket).
TW4	Twente	Accelerometer, Gyroscope.	4	3	6	Walk, Sit, Run, Downstair, Stand, Upstair (Pocket).
TW5	Twente	Accelerometer, Gyroscope, Magnetometer.	4	3	6	Walk, Sit, Run, Downstair, Stand, Upstair (Pocket).
IH1	In-House	GPS, Accelerometer, Gyroscope, Magnetometer.	7	4	5	Car, Stop, Walk, Bike, Run.
IH2	In-House	GPS, Accelerometer, Gyroscope, Magnetometer.	1	1	3	Car, Stop, Walk.

IH3	In-House	GPS, Accelerometer, Gyroscope, Magnetometer.	1	1	3	Car, Stop, Walk.
IH4	In-House	GPS, Accelerometer, Gyroscope, Magnetometer.	1	1	3	Car, Stop, Walk.
IH5	In-House	GPS, Accelerometer, Gyroscope, Magnetometer.	1	1	3	Car, Stop, Walk.
IH6	In-House	GPS, Accelerometer, Gyroscope, Magnetometer.	1	1	3	Car, Stop, Walk.
IH7	In-House	GPS, Accelerometer, Gyroscope, Magnetometer.	1	1	4	Car, Stop, Walk, Bike.
IH8	In-House	GPS, Accelerometer, Gyroscope, Magnetometer.	1	1	3	Run, Stop, Walk.

Tabella 3.2: Descrizione dei dataset utilizzati nella fase di test del software.

Classificatori									
Tipologia		Regole		Alberi		Bayesiani		Reti Neurali	SVM
Feat. Set	Dataset	OneR	J48	Ran.Forest	Nai.Bayes	BayesNet	Multi.Percep.	LibSVM	
1	IH1	81,58%	81,97%	82,04%	81,28%	81,44%	81,27%	81,55%	Migliore
2	IH1	81,37%	82,01%	81,76%	67,68%	81,45%	81,15%	81,56%	Ran.Forest
3	IH1	81,37%	82,78%	83,02%	67,02%	81,51%	81,14%	82,41%	J48
4	IH1	81,37%	82,77%	82,98%	66,37%	76,78%	81,25%	81,45%	Ran.Forest
5	IH1	81,37%	86,59%	87,14%	73,10%	83,45%	85,13%	83,46%	Ran.Forest
6	IH1	81,37%	86,51%	87,23%	81,10%	83,12%	85,05%	83,14%	Ran.Forest
7	IH1	81,37%	87,76%	88,59%	83,25%	84,84%	86,24%	84,20%	Ran.Forest
8	IH1	81,37%	91,36%	91,08%	84,05%	85,81%	87,28%	84,46%	J48
9	IH1	81,37%	86,88%	86,55%	80,46%	84,28%	86,02%	84,41%	J48
10	IH1	81,37%	87,68%	87,69%	81,32%	81,96%	84,43%	83,81%	Ran.Forest
11	IH1	77,65%	79,01%	78,91%	74,30%	72,89%	77,17%	77,76%	J48
12	IH1	78,10%	80,69%	80,06%	70,31%	78,44%	79,47%	75,95%	J48
13	IH1	71,36%	76,82%	74,99%	68,76%	71,67%	72,58%	77,89%	J48
14	IH1	77,65%	83,96%	84,10%	76,15%	77,68%	81,06%	80,49%	Ran.Forest
15	IH1	77,65%	87,12%	88,49%	77,50%	80,11%	82,89%	83,03%	Ran.Forest

Tabella 3.3: Performance dei classificatori per ciascun feature set.

Classificatore				
Tipologia		Alberi		Migliore
Feature Set	Dataset	J48	RandomForest	
3	GL1	70,16%	64,74%	J48
3	GL2	91,80%	91,24%	J48
3	GL3	77,53%	73,89%	J48
11	TW1	47,91%	41,30%	J48
12	TW2	50,82%	44,51%	J48
13	TW3	61,38%	53,68%	J48
14	TW4	54,92%	52,02%	J48
15	TW5	68,20%	70,04%	Ran.Forest
8	IH1	91,36%	91,08%	J48
4	IH2	88,26%	88,75%	Ran.Forest
5	IH2	89,88%	89,14%	J48
7	IH2	89,12%	89,56%	Ran.Forest
8	IH2	90,43%	92,05%	Ran.Forest
8	IH3	92,34%	91,47%	J48
8	IH4	93,07%	94,55%	Ran.Forest
8	IH5	95,87%	96,56%	Ran.Forest
8	IH6	96,50%	96,37%	J48
8	IH7	93,65%	94,16%	Ran.Forest
8	IH8	95,42%	96,48%	Ran.Forest

Tabella 3.4: Performance dei migliori classificatori per ciascun dataset.

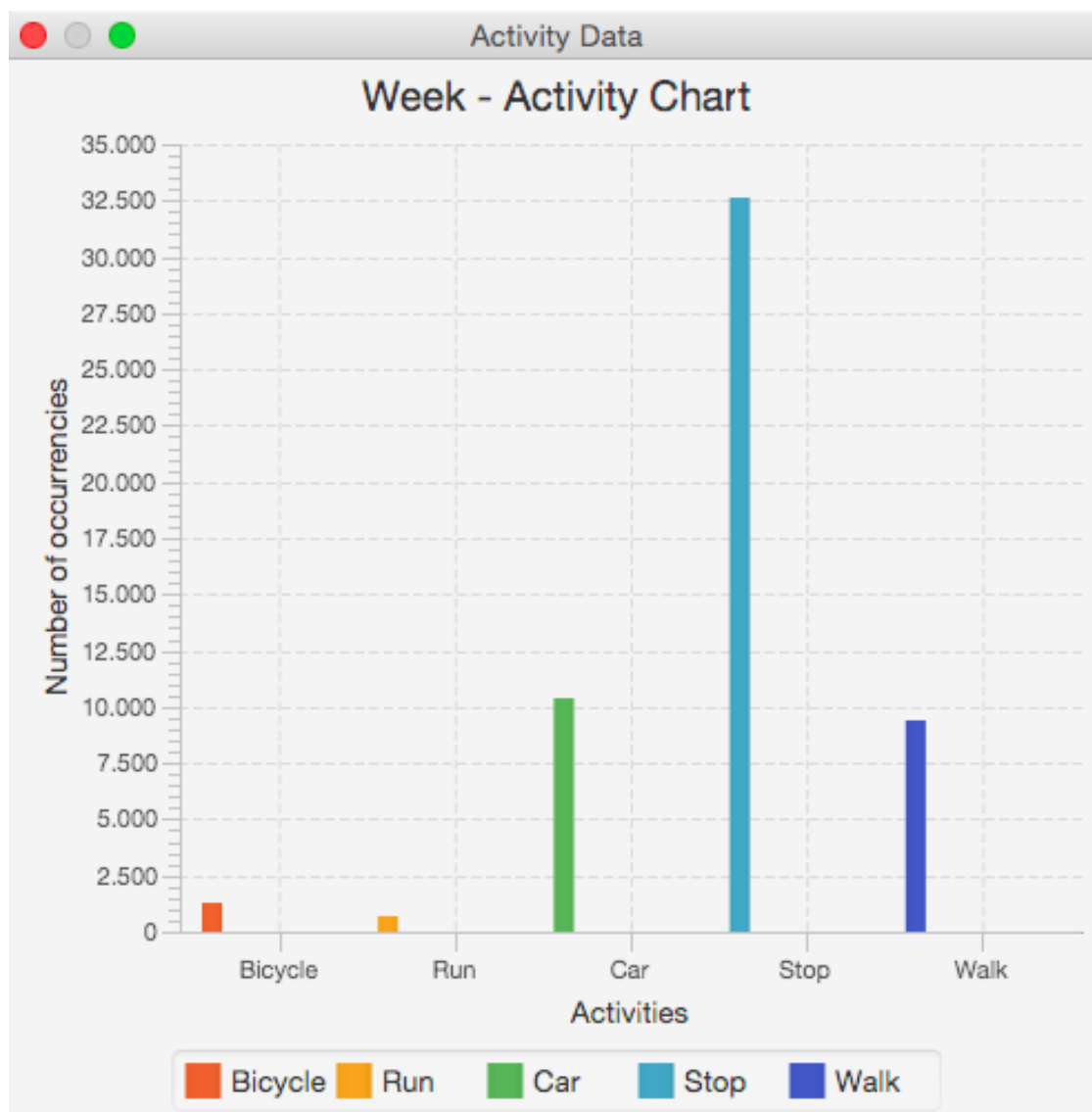


Figura 3.10: Grafico con la distribuzione dei mezzi di trasporto del dataset utilizzato per individuare il classificatore migliore.

Capitolo 4

Clustering

Scopo di questo capitolo è descrivere tutte le operazioni eseguite sui dati grezzi e ciascun algoritmo applicato ai fini di effettuare il riconoscimento del mezzo di trasporto utilizzato dagli utenti mediante un approccio non supervisionato di tipo non parametrico, sfruttando gli algoritmi di clustering. Nella sezione 4.1 viene presentato l'argomento e vengono descritti i concetti di base, mentre nella 4.2 vengono illustrati tutti i passaggi eseguiti sui dati per giungere al risultato finale. Nelle sezioni 4.3, 4.4 e 4.5 si entra nel dettaglio degli algoritmi utilizzati ed infine, nella sezione 4.6, vengono discussi i risultati del processo.

4.1 Concetti di base

Le tecniche di clustering (si vedano [1] e [24]) si applicano ad un dataset quando non esiste un attributo nominale che partiziona le istanze in un insieme di classi, come nel caso della classificazione. Si cerca di trovare possibili raggruppamenti in cui le istanze stesse possono essere suddivise, osservando solamente le loro caratteristiche, ovvero i valori degli attributi. Questi raggruppamenti riflettono meccanismi che caratterizzano il dominio da cui provengono i dati, i quali fanno in modo che alcune istanze risultino simili ad altre più di quanto quelle rimanenti non lo siano. L'essenza del clustering, dunque, consiste nel trovare tecniche per poter verificare tale similarità tra istanze per poterle poi raggruppare.

Il risultato degli algoritmi di clustering può essere di diverso tipo:

- *esclusivo*: se un'istanza può appartenere solamente ad un solo cluster;
- *sovrapposto*: se un'istanza può appartenere contemporaneamente a più cluster;

- *probabilistico*: se un'istanza appartiene a diversi cluster secondo determinate probabilità.

Bisognerebbe scegliere, dunque, l'algoritmo che meglio rappresenta il meccanismo che descrive il dataset analizzato. Tuttavia, nella maggior parte dei casi tale meccanismo è sconosciuto, perciò la scelta non è mai così semplice. Le tecniche di clustering non nascono per raggruppare dati esclusivamente numerici; vengono utilizzate, infatti, anche nell'analisi dei testi; alcuni algoritmi che trattano proprio dati numerici, inoltre, sfruttano concetti nati con l'analisi dei testi stessa, come si vedrà nelle sezioni seguenti; è opportuno, per tale motivo, introdurre alcuni termini utili. Con *documento* si intende un insieme di istanze del dataset da analizzare; ognuno di essi è composto da *words*, o parole, che sono le singole istanze del dataset. Ogni documento è generato da uno o più topic, o argomenti, che corrispondono ai cluster che verranno individuati una volta applicato l'algoritmo. Infine, la *topic mixture* è l'insieme degli argomenti a cui il documento appartiene con la relativa percentuale. Ricapitolando, diversi algoritmi di clustering svolgono il loro compito rappresentando i dati da analizzare secondo un *topic model*, un modello statistico per scoprire gli argomenti "astratti" presenti in una collezione di documenti, dove tali argomenti sono i cluster in cui raggruppare le istanze. poiché un singolo documento può appartenere a più argomenti, si tratta di una forma di clustering con sovrapposizione.

A questo è evidente perché il clustering è un approccio non supervisionato; non esiste, infatti, una fase di costruzione di un determinato modello per guidare l'azione dell'algoritmo sulla base di un training set. Rimane da chiarire cosa si intenda parlando di versione non parametrica di questo approccio non supervisionato, ovvero l'obiettivo della seconda parte di questo lavoro, come descritto nella sezione 1.2.

Tale caratteristica consiste nel fatto che molti algoritmi di clustering chiedono, in input, la specifica del numero di cluster da individuare. In questo caso, tale categoria di algoritmi non è utilizzabile; come si può prevedere il numero di mezzi di trasporto utilizzati dagli utenti se il punto è proprio scoprirli? si parla di approccio non parametrico e non supervisionato, dunque, perché lo scopo è utilizzare algoritmi di clustering che non richiedono tale parametro in anticipo, poiché sono in grado di determinarlo autonomamente per poi eseguire il raggruppamento delle istanze senza sfruttare gli esempi di un training set. Al modulo del software che gestisce il clustering stesso, dunque, viene presentato il dataset da analizzare, e l'output sono i cluster riconosciuti, che in questo particolare caso rappresentano i mezzi di trasporto utilizzati dall'utente.

4.2 Operazioni sui dati

Per realizzare l'approccio non parametrico e non supervisionato di cui si è parlato nella sezione precedente vengono usati più algoritmi di clustering e i dati sono sottoposti a diversi trattamenti. Nel seguito, dunque, si intende presentare il flusso di lavoro, per poi entrare nel dettaglio di tutte le operazioni eseguite. Il processo si avvia dal menù apposito, una volta ottenuta durante l'utilizzo della finestra dell'interfaccia in figura 2.9.

Nel processo che porta al riconoscimento dei mezzi di trasporto sfruttando questo approccio non parametrico, il punto di partenza sono i feature vector. Come per la classificazione, risulta utile ridurre la dimensione dei dati grezzi da analizzare e il numero di attributi che li descrivono calcolando le feature; nelle prime implementazioni del processo di clustering all'interno del software esso veniva applicato ai dati grezzi, senza passare per il calcolo dei feature vector. Il tempo di esecuzione del processo risultava essere molto elevato e costituiva un notevole collo di bottiglia nell'elaborazione. Nell'implementazione attuale, invece, applicando l'algoritmo sui feature vector, si è percepita ad occhio nudo una considerevole riduzione del tempo d'esecuzione, a dimostrazione che non si tratta di un'operazione inutile. Nella figura 4.1, dunque, si può osservare l'albero delle chiamate ai metodi che realizzano il processo non parametrico di riconoscimento. Il nodo contenente il metodo *cluster* ha due figli poiché è possibile tra due alternative dello stesso algoritmo descritto nella sezione 4.3.

Il metodo *doClustering* si limita a preparare l'interfaccia grafica che visualizzerà grafici e pulsanti per esplorare i risultati finali e le fasi intermedie del processo. Quello chiamato successivamente, *cluster*, riceve in input tre parametri; il numero massimo di iterazioni, la dimensione della *finestra scorrevole* e la dimensione della *sovrapposizione*; l'utilità di essi diventerà chiara nel seguito. Il primo passaggio eseguito da tale metodo consiste nel prendere i feature vector di ciascun sensore ed inserirli in altrettante strutture dati, prerequisito per la fase successiva. L'aspetto interessante sta nel fatto che lo spazio delle feature viene suddiviso in al massimo quattro sottoinsiemi, uno per sensore (tre o meno se uno o più sensori non sono stati utilizzati nella produzione dei feature vector). A questo punto, su ciascuno dei sottoinsiemi di feature descritte dai rispettivi feature vector memorizzati nelle strutture dati, viene chiamato il primo algoritmo di clustering utilizzato, il *Dirichlet Process Mixture Model* (descritto nella sezione 4.3). L'utente, utilizzando l'interfaccia grafica, può scegliere se attivare il *Gaussian Dirichlet Process Mixture Model* oppure il *Multinomial Dirichlet Process Mixture Model*, due diverse implementazioni del DPMM. Questo algoritmo di clustering raggruppa i feature vector in cluster, assegnando a ciascuno di essi un'etichetta, ed itera al massimo per il numero di iterazioni passato come parametro. Si ottengono, dunque, da uno a quattro

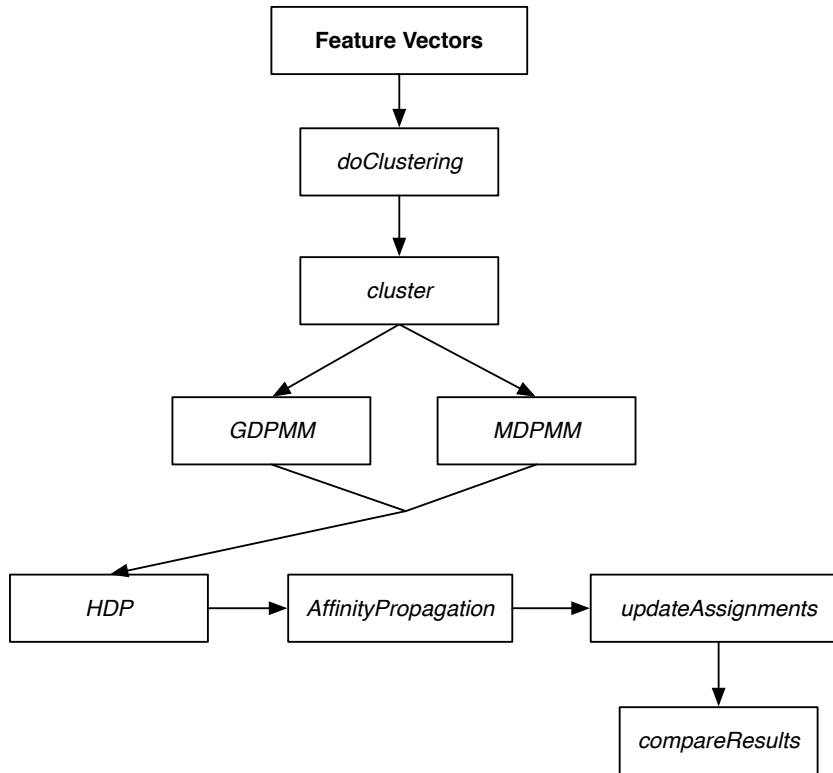


Figura 4.1: Schema delle chiamate effettuate nel processo non parametrico di riconoscimento.

sottoinsiemi di feature vector raggruppati in cluster, derivanti da altrettante esecuzioni del DPMM, dove ogni cluster di ciascun sottoinsieme possiede un'etichetta; tale operazione viene eseguita per evitare i problemi di Overfitting già descritti nella sezione 3.3. A questo punto, per ciascun feature vector, si copiano le etichette ottenute in ciascuna esecuzione del DPMM in un vettore ausiliario, ottenendo così la *parola artificiale* descrivente il feature vector stesso; le parole artificiali sono tante quanti sono i feature vector stessi e vengono memorizzate in una matrice apposita. In figura 4.2 è possibile osservare uno schema del processo di clustering fino a questo punto. Nelle figure 4.3 e 4.4 sono presenti un esempio di esecuzione del GDPMM sui quattro sottoinsiemi di feature con le relative parole artificiali. L'idea che sottende la creazione di tali parole artificiali consiste nel fatto che, per esempio, se le feature di GPS, accelerometro e giroscopio assegnano al feature vector uno l'etichetta del cluster due, mentre il magnetometro assegna l'etichetta del cluster uno, allora il feature vector globale fa parte del cluster due, con una buona probabilità. C'è incertezza nell'assegnazione solamente quando tutte le etichette sono diverse.

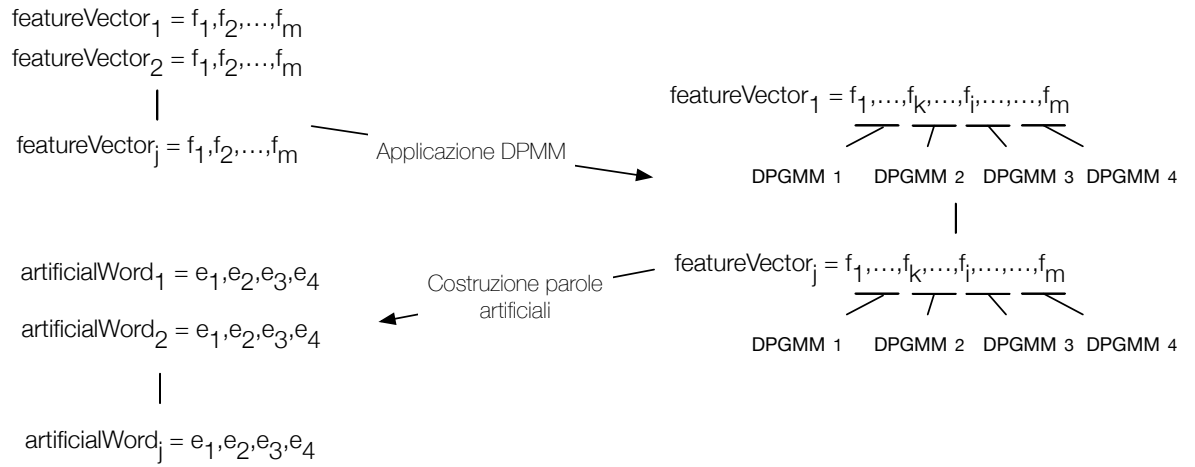
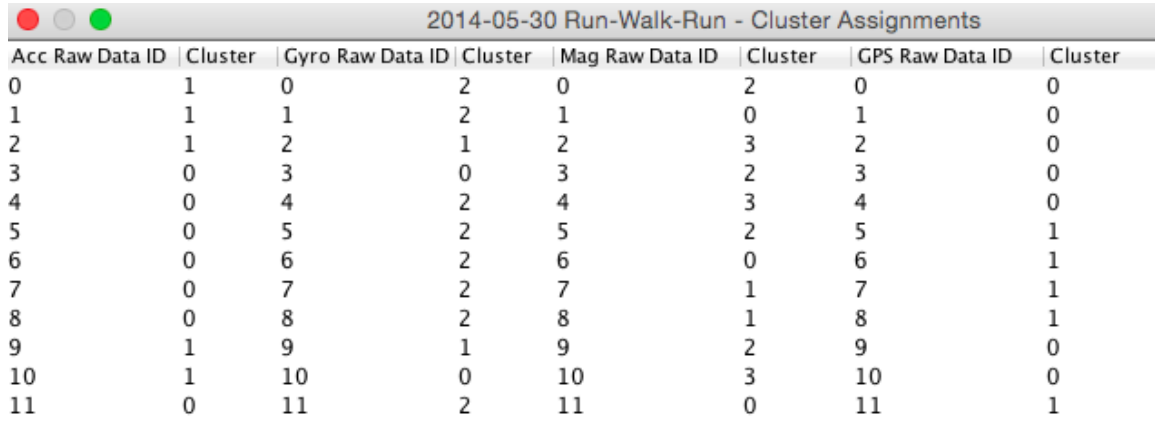


Figura 4.2: Schema descrivente il primo passaggio del processo di clustering.

A questo punto, la matrice delle parole artificiali viene passata in input al metodo successivo chiamato *HDP*, assieme alla dimensione della finestra scorrevole ed alla sovrapposizione. Questo metodo, come primo passaggio, esegue la creazione dei documenti; un documento è costituito da un insieme di parole artificiali ed il loro numero è dato dalla dimensione della finestra scorrevole; il fattore di sovrapposizione indica il numero di parole artificiali che due documenti successivi hanno in comune. Per esempio, si supponga di avere sessanta secondi di registrazione, con un intervallo temporale di calcolo delle feature pari a dieci secondi. I feature vector corrispondenti sono sei; dopo l'applicazione del DPMM si hanno altrettante parole artificiali. Si supponga che la dimensione della finestra scorrevole sia pari a tre, ed il fattore di sovrapposizione ad uno; dopo l'applicazione della prima parte del metodo, si ottengono tre documenti; il primo avrà i feature vector uno, due e tre; il secondo i feature vector tre, quattro e cinque; il terzo, avrà i feature vector cinque e sei. I primi due, quindi, riassumeranno trenta secondi di registrazione, con una sovrapposizione di dieci secondi per il secondo, mentre il terzo ne riassumerà venti, con dieci secondi di sovrapposizione. A questo punto, si ha l'intero vocabolario rappresentato come un flusso di tali parole artificiali suddivise in diversi documenti (come si può notare, emergono i termini descritti nella sezione 4.1). Il passaggio successivo consiste nell'eseguire il secondo algoritmo di clustering per ciascun documento così creato. Tale algoritmo è chiamato *Hierarchical Dirichlet Process*, da cui il nome nel metodo (si veda la sezione 4.4). Ciò che fa, sostanzialmente, consiste nel raggruppare le parole artificiali in un ulteriore insieme di cluster; una volta fatto ciò, viene calcolata la topic mixture, dividendo la frequenza cumulata

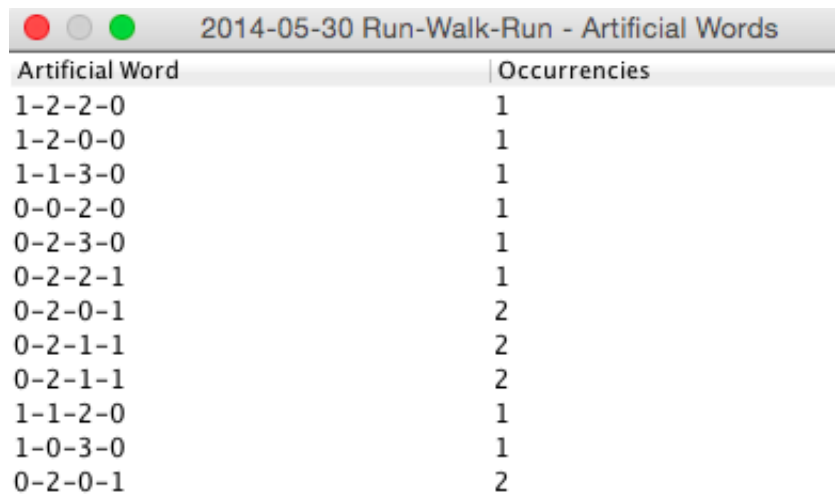


Acc Raw Data ID	Cluster	Gyro Raw Data ID	Cluster	Mag Raw Data ID	Cluster	GPS Raw Data ID	Cluster
0	1	0	2	0	2	0	0
1	1	1	2	1	0	1	0
2	1	2	1	2	3	2	0
3	0	3	0	3	2	3	0
4	0	4	2	4	3	4	0
5	0	5	2	5	2	5	1
6	0	6	2	6	0	6	1
7	0	7	2	7	1	7	1
8	0	8	2	8	1	8	1
9	1	9	1	9	2	9	0
10	1	10	0	10	3	10	0
11	0	11	2	11	0	11	1

Figura 4.3: Esempio di esito delle esecuzioni del GDPMM.

delle parole assegnate a ciascun cluster per il numero totale di esse. In figura 4.6 è possibile osservare un esempio di topic mixture individuate a partire da un insieme di tre documenti.

Le operazioni descritte nella fase precedente vengono svolte per un motivo particolare; si supponga che, per esempio, durante una determinata sessione di registrazione, sia stato fatto un giro in bicicletta, seguito da una camminata, e da un altro giro con la stessa bicicletta. Il DPMM si limiterebbe ad individuare tre cluster, bicicletta-camminata-bicicletta e di conseguenza tre mezzi di trasporto diversi, nonostante sia possibile che il mezzo utilizzato nel primo e nel terzo cluster sia lo stesso. Qui entrano in gioco le topic mixture di ciascun documento; se sono simili, vuol dire che il mezzo utilizzato è effettivamente lo stesso; quindi, nell'esempio di poco fa, i mezzi di trasporto veramente utilizzati sono due. Naturalmente, è possibile anche che l'utente abbia cambiato bicicletta, dopo la camminata, oppure che abbia pedalato molto più velocemente rispetto a prima; in questo caso, le topic mixture non risulteranno simili e i mezzi individuati saranno effettivamente tre. Per far ciò, le topic mixture stesse vengono passate in input ad un terzo metodo, *affinityPropagation*; tale metodo applica un ulteriore algoritmo di clustering, chiamato proprio Affinity Propagation (si veda la sezione 4.5), con lo scopo di raggruppare proprio quelle topic mixture che risultano simili. Ogni cluster di topic mixture individuato, dunque, corrisponde ad un mezzo di trasporto, dove esso occorre una o più volte. Riassumendo, ciò che l'algoritmo individua non sono necessariamente n mezzi di trasporto distinti poiché lo stesso mezzo potrebbe essere stato utilizzato in maniera diversa in due insiemi di rilevazioni, quindi tali insiemi vengono separati dall'algoritmo stesso. Tornando all'esempio precedente, è possibile riconoscere gli insiemi “bicicletta uno” e “bicicletta due”, tuttavia, il mezzo



Artificial Word	Occurrences
1-2-2-0	1
1-2-0-0	1
1-1-3-0	1
0-0-2-0	1
0-2-3-0	1
0-2-2-1	1
0-2-0-1	2
0-2-1-1	2
0-2-1-1	2
1-1-2-0	1
1-0-3-0	1
0-2-0-1	2

Figura 4.4: Esempio di parole artificiali generate a partire dall'esito del DPMM.

utilizzato è comunque una bicicletta; ai fini del riconoscimento, dunque, i dati grezzi dei due insiemi ricevono tale etichetta. Quanto svolto da questo metodo è schematizzato in figura 4.7.

La quarta fase viene svolta dal metodo *updateAssignments*; esso riceve in input gli assegnamenti effettuati da Affinity Propagation ed ha il compito di etichettare i dati grezzi sulla base del mezzo di trasporto a cui sono stati assegnati. Ricapitolando, un mezzo di trasporto è un cluster di topic mixture ed ognuna di esse descrive la struttura di un determinato documento. Un documento è composto da diverse parole artificiali, ed ogni parola artificiale descrive il lavoro fatto durante la prima fase su un singolo feature vector. Ogni feature vector riassume diversi dati grezzi. Seguendo tale percorso inverso, dunque, si può giungere all'etichetta finale da assegnare al singolo dato grezzo, partendo dai documenti raggruppati in base alle loro topic mixture. Si veda lo schema in figura 4.8.

La quinta ed ultima fase consiste nella chiamata al metodo *compareResults*. Come può suggerire il suo nome, lo scopo di esso consiste nel valutare i risultati del processo di clustering. Esso riceve in input il vettore contenente l'etichetta finale di ciascun dato grezzo e quello contenente le etichette originali e, sfruttando tali informazioni, costruisce la struttura di ciascun mezzo di trasporto individuato memorizzandola, infine, in una struttura dati di supporto. Ricapitolando, quello che Affinity Propagation restituisce in output è un vettore di numeri, dove ciascun numero è l'etichetta di un documento. Tutti i

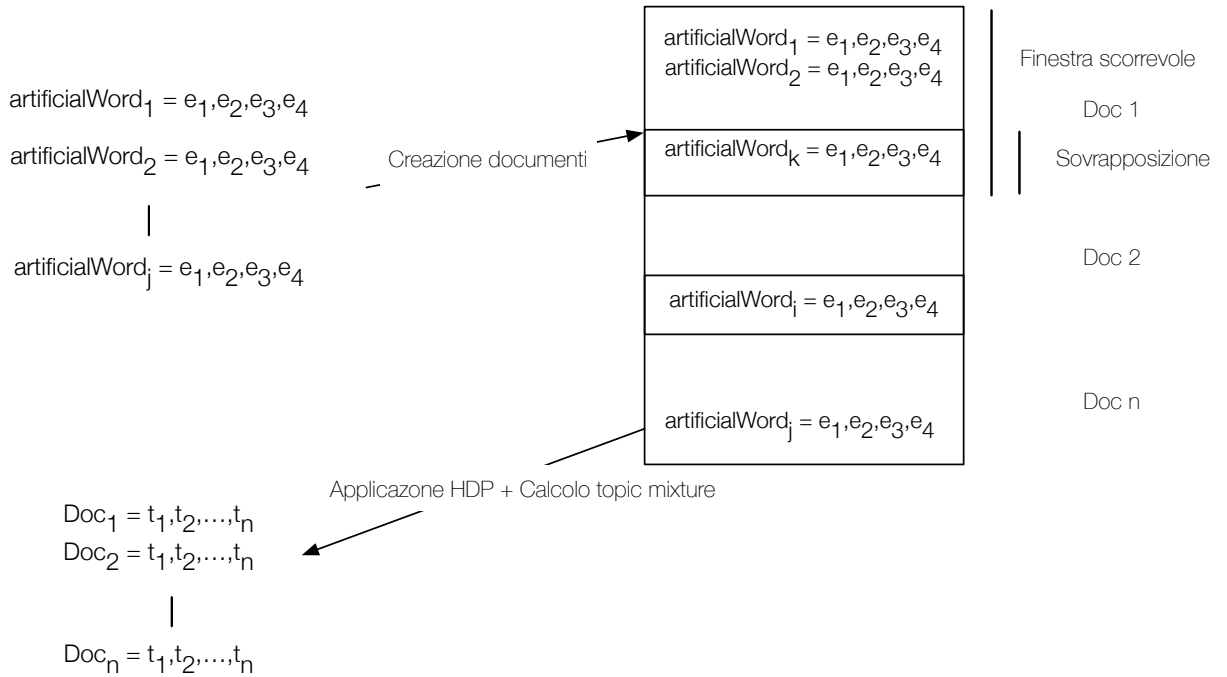
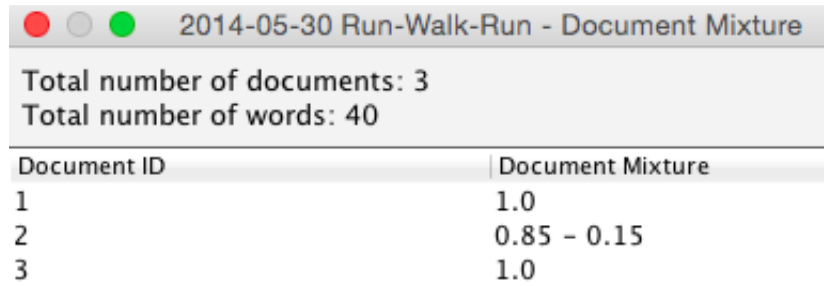


Figura 4.5: Schema descrivente il secondo passaggio del processo di clustering.

documenti con la stessa etichetta indicano un mezzo di trasporto. Quello che fa `updateAssignments`, invece, consiste nell'assegnare l'etichetta dei documenti stessi ai dati grezzi che riassumono compiendo il percorso inverso precedentemente descritto. La struttura di ciascun mezzo di trasporto, dunque, viene costruita trasformando l'etichetta numerica derivata dall'esecuzione di AP in una che indichi un mezzo vero e proprio. Per esempio, il mezzo di trasporto zero non sarà più un semplice insieme di dati grezzi, ma sarà costituito da trentasette dati di tipo "run" e da cinquantadue dati di tipo "car"; si veda la figura 4.10 che contiene un esempio di tale operazione. Per far ciò, tuttavia, è necessario un metodo che analizzi le caratteristiche dei dati grezzi contenuti in ciascun mezzo di trasporto individuato per poi stabilire di che tipo di mezzo si tratti. Tale metodo non è stato implementato poiché è sufficiente eseguire un'approssimazione dei risultati, per testare la bontà del lavoro; dato che il dataset a disposizione è etichettato, il singolo dato grezzo contiene anche l'attributo nominale indicante il mezzo di trasporto a cui fa riferimento in origine, sfruttato nella classificazione; l'operazione di traduzione da etichette numeriche a etichette reali viene svolta analizzando contemporaneamente il vettore originale dei dati grezzi e quello degli assegnamenti finali; per esempio, il dato grezzo uno, che nel vettore finale fa parte del mezzo di trasporto zero, in quello delle etichette originali è un dato di tipo "run" (si veda lo schema in figura



Document ID	Document Mixture
1	1.0
2	0.85 - 0.15
3	1.0

Figura 4.6: Esempio di topic mixture generate a partire dall'esito dell'HDP.

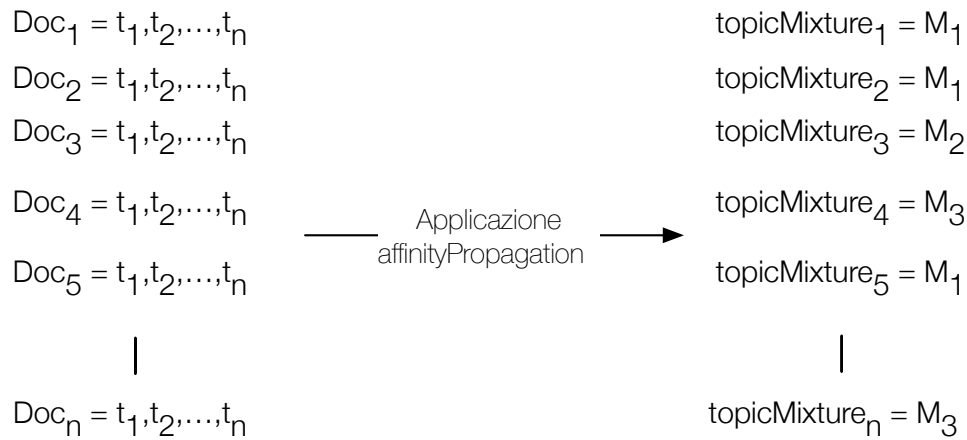


Figura 4.7: Schema descrivente il terzo passaggio del processo di clustering.

4.9). La struttura del mezzo di trasporto, dunque, viene costruita contando le occorrenze di ciascuna etichetta vera per ognuna di quelle individuate dopo l'esecuzione di AP, nell'insieme costituito da tutti i dati grezzi. Prima di tradurre definitivamente le etichette numeriche in etichette vere, viene fatta un'ulteriore considerazione; a tutti i dati grezzi che costituiscono un mezzo di trasporto individuato dall'AP, viene assegnata l'etichetta reale più frequente nella loro struttura; facendo riferimento nuovamente alla figura 4.11, per esempio, tutti i dati appartenenti al mezzo di trasporto zero saranno di tipo "walk". La motivazione che sottende questo passaggio è che un mezzo di trasporto non può essere contemporaneamente di due tipi. Si precisa nuovamente che queste considerazioni finali servono per testare i risultati finali normalizzandoli con una buona approssimazione. A questo punto, una volta eseguita l'operazione di traduzione, vengono calcolate le percentuali di insuccesso e successo del processo di clustering, osservando nuovamente i dati originali. Per poter

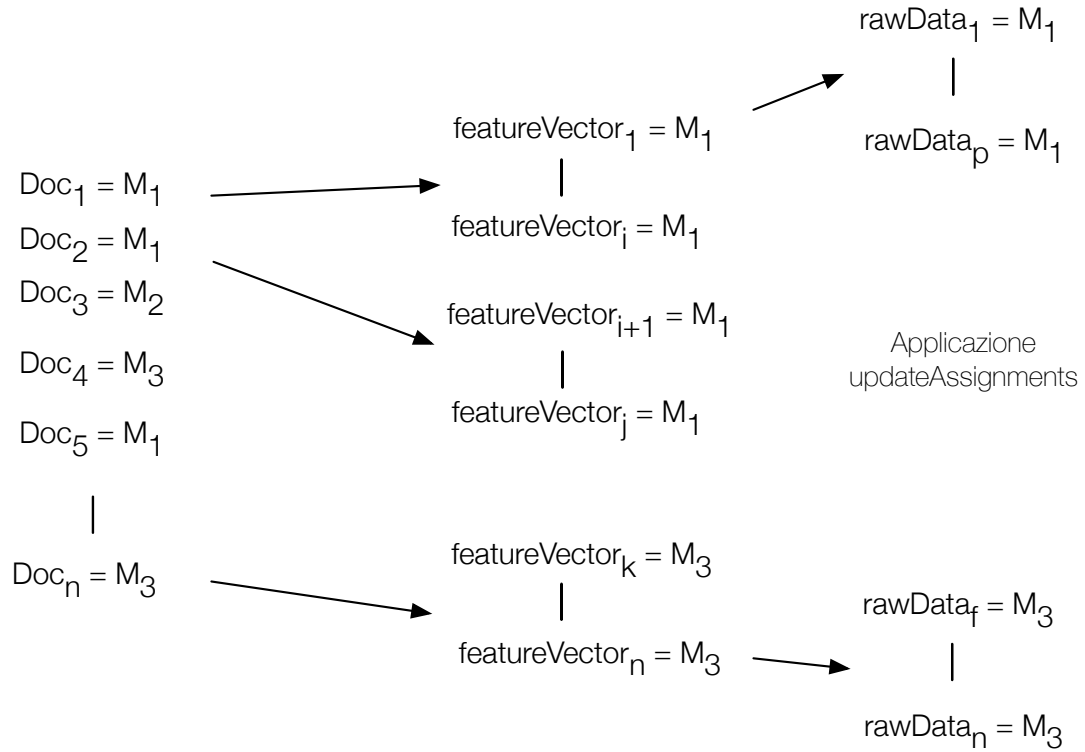


Figura 4.8: Schema descrivente il quarto passaggio del processo di clustering.

trasformare questo lavoro in un prodotto utilizzabile da una platea di utenti ed eliminare la componente etichettata, dunque, è necessario eliminare tale approssimazione implementando un metodo per etichettare i dati raggruppati sulla base delle loro caratteristiche, senza il supporto di quelli originali. In figura 4.11 è possibile osservare una piccola parte dell'esito di una esecuzione del metodo in oggetto.

Una volta terminate le fasi precedentemente descritte, viene costruita una nuova finestra nella GUI e gli assegnamenti finali calcolati dal processo di clustering vengono confrontati visivamente con quelli originali sfruttando tre diverse tipologie di grafico. Il primo è quello della figura 4.12, che mostra sull'asse delle ordinate l'etichetta assegnata ai dati grezzi dopo l'esecuzione di AP, mentre sull'asse delle ascisse vengono indicati gli identificatori dei dati grezzi. Il secondo grafico è simile al primo, e si può osservare in figura 4.13; la differenza sta nel fatto che nell'asse delle ordinate vengono inserite le etichette originali dei dati grezzi. Il terzo grafico, invece, mostra due serie diverse di dati; una contiene i dati finali etichettati con le etichette reali mentre la seconda mostra nuovamente quelli originali. In tal modo, si può ottenere un utile



Figura 4.9: Schema descrivente il quinto passaggio del processo di clustering.

feedback visivo relativamente ai punti in cui il processo sbaglia; un esempio di tale grafico è presente in figura 4.14. Se è visibile solamente la linea blu, allora tutte le istanze del dataset sono state raggruppate correttamente.

Nella prima tab della nuova finestra dell'interfaccia, inoltre, sono presenti da uno a quattro ulteriori grafici (a seconda dei sensori scelti all'inizio del processo) relativi agli esiti delle esecuzioni del DPMM. Nella parte inferiore della finestra stessa sono disponibili diversi pulsanti per analizzare diverse caratteristiche dei risultati ottenuti; si veda la figura 4.15. Con questo ultimo passaggio, dunque, termina il lavoro effettuato per implementare tale approccio non supervisionato di tipo parametrico; nelle sezioni seguenti viene descritto più approfonditamente il funzionamento dei tre algoritmi di clustering chiamati precedentemente in causa e successivamente lasciati in sospeso.

4.3 Dirichlet Process Mixture Model

Il primo algoritmo di clustering utilizzato durante il processo di riconoscimento non supervisionato e non parametrico è il DPMM, acronimo di *Dirichlet Process Mixture Model*. Questo algoritmo viene realizzato in due versioni diverse chiamate GDPMM, ovvero *Gaussian Dirichlet Process Mixture Model* e MDPMM, che sta per *Multinomial Dirichlet Process Mixture Model*. In [13] è disponibile un'introduzione all'algoritmo, mentre in [9], [10] e [11] vengono descritte approfonditamente le varie parti di cui si compone, entrando nel dettaglio sugli aspetti matematici; vengono trattate anche le due varianti, gaussiana e multinomiale. In [12], invece, viene resa disponibile un'implementazione in Java, sfruttata in questo lavoro. Come è stato precisato nella sezione

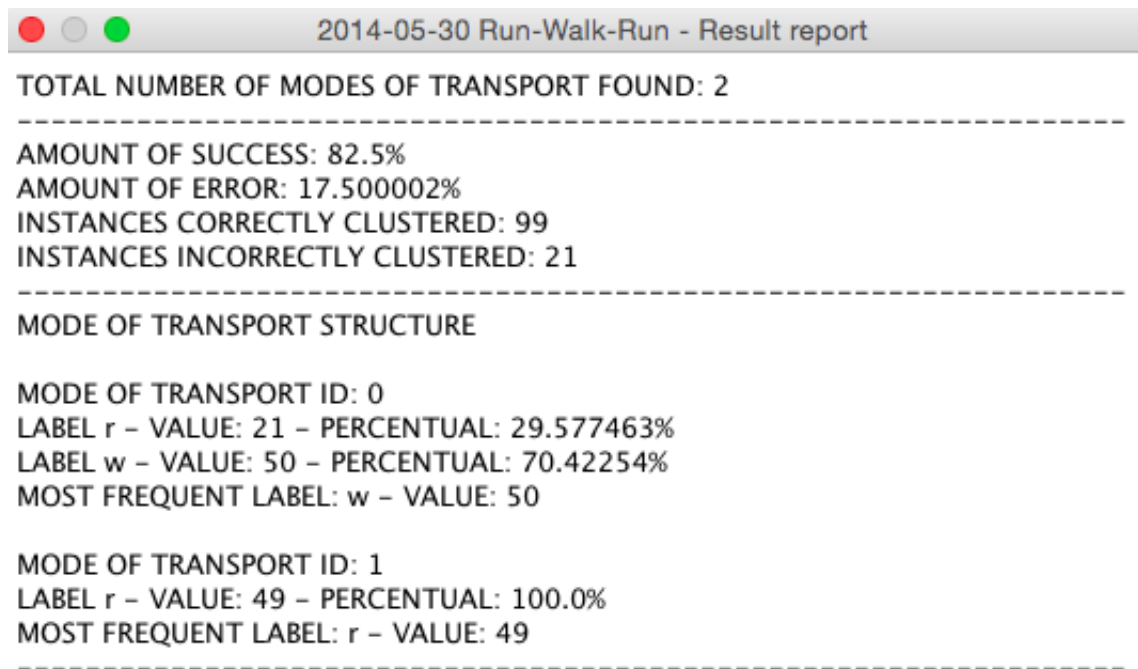
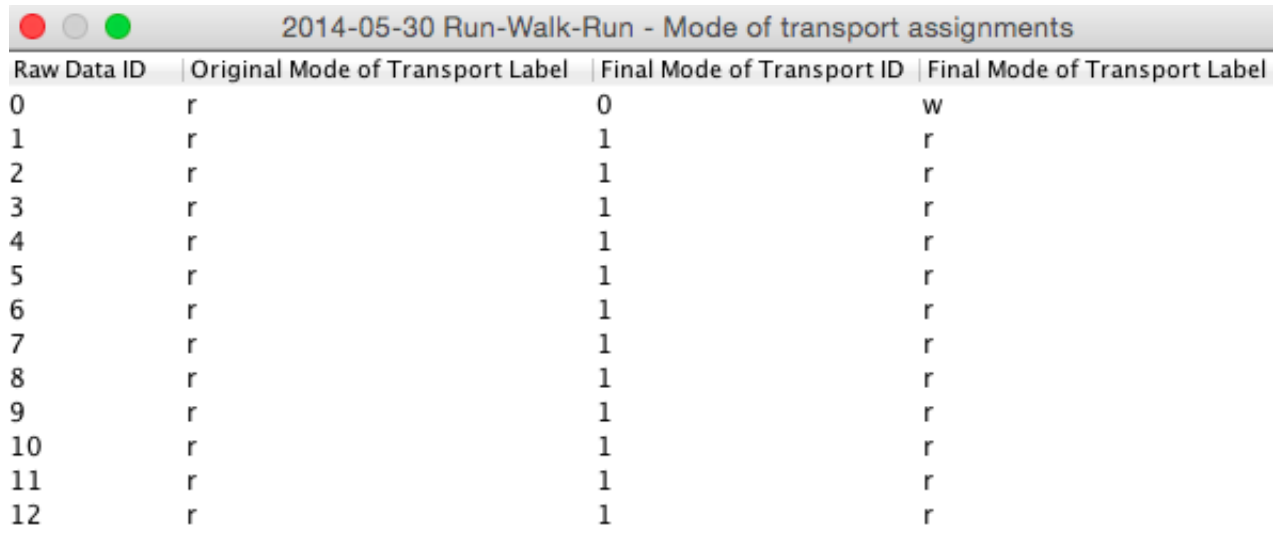


Figura 4.10: Esempio di rapporto finale sui risultati ottenuti dal processo di clustering.

4.2, il dataset ricevuto in input dal DPMM è l'insieme dei feature vector. La domanda chiave a cui esso cerca di rispondere è: com'è possibile utilizzare i dati raccolti per scoprire differenti tipi di gruppi? algoritmi classici come LDA o K-Means assumono che ci sia un numero prefissato di cluster da trovare; ma per molti dataset reali non è possibile specificare in anticipo tale parametro. Chi può assumere in quanti modi è possibile raggruppare i dati relativi ai sensori di cui tanto si è discusso fino ad ora? quello che il DPMM fa è supporre che esistano inizialmente infiniti cluster possibili; il numero di quelli effettivamente individuati crescerà all'aumentare dei dati ricevuti. Esso, dunque, è un *algoritmo di clustering non parametrico bayesiano*. Tale definizione indica una classe di tecniche dove dei parametri possono cambiare secondo la natura dei dati. In questo caso, il numero di cluster.

Il DPMM ha bisogno di un modo per descrivere ciascun cluster; avendo a che fare con dati numerici continui, il modello generativo di essi è una distribuzione di probabilità gaussiana D_0 , caratterizzata da una media u_i e da una varianza σ_i . Un dato tratto da un cluster, dunque, è un campione proveniente dalla distribuzione che caratterizza il cluster stesso.

L'assegnamento dei feature vector a ciascun cluster viene effettuato secondo un algoritmo chiamato CRP ([13]), ovvero *Chinese Restaurant Process*. Il nome



Raw Data ID	Original Mode of Transport Label	Final Mode of Transport ID	Final Mode of Transport Label
0	r	0	w
1	r	1	r
2	r	1	r
3	r	1	r
4	r	1	r
5	r	1	r
6	r	1	r
7	r	1	r
8	r	1	r
9	r	1	r
10	r	1	r
11	r	1	r
12	r	1	r

Figura 4.11: Esempio di dati raggruppati tradotti con etichette reali secondo l'approssimazione descritta.

deriva dall'analogia utilizzata per spiegare il suo funzionamento; si immagini, infatti, di andare con un gruppo di amici al ristorante:

- Inizialmente il ristorante è vuoto
- La prima persona ad entrare, chiamata Alan, si siede ad un tavolo (il dato grezzo viene inserito in un cluster) e ordina del cibo (vengono impostati i parametri della distribuzione descrivente il cluster stesso): le altre persone che si siederanno al tavolo dovranno accontentarsi e mangiare il cibo che ha ordinato Alan.
- Una seconda persona, chiamata Andrea, si siede ad un nuovo tavolo con probabilità $a/(1+a)$ (il feature vector viene inserito in un nuovo cluster), mentre si siede allo stesso di Alan con probabilità $1/(1+a)$ e mangia il cibo ordinato da quest'ultimo (il feature vector viene inserito nello stesso cluster).
- ...
- La $n+1$ -esima persona che entra nel ristorante si siede ad un nuovo tavolo con probabilità $a/(n+a)$, ad uno specifico tavolo t con probabilità $n_t/(n+a)$, dove n_t è il numero di persone correntemente sedute a tale tavolo.

A questo punto si possono fare le seguenti considerazioni:

- Più persone (dati grezzi) si siedono allo stesso tavolo (vengono assegnati allo stesso cluster), più diventa probabile che nuove persone si uniscano ad esse.
- C'è sempre una piccola probabilità che qualcuno si sieda ad un tavolo vuoto.
- Tale probabilità dipende da a ; tale valore, dunque, può essere visto come un *parametro di dispersione* che influenza, appunto, la dispersione dei dati grezzi nei cluster. Più a è piccolo, meno sono i cluster creati dal CRP, più esso è grande più sono i cluster creati.

A questo punto, si immagini di eseguire il CRP per un numero predefinito di volte. Ciascuna iterazione di tale algoritmo genera una distribuzione di persone ai tavoli (feature vector nei cluster) e ciascuna distribuzione sarà diversa da quella dell'iterazione precedente. Per esempio, il cluster uno, inizialmente, potrebbe contenere il venti per cento dei dati, successivamente il tredici per cento, ecc. Quello che si ottiene, dunque, eseguendo diverse volte tale processo è una distribuzione di distribuzioni di persone ai tavoli. Si arriva, dunque, a definire un processo di Dirichlet.

Un *Processo di Dirichlet*, date una distribuzione base D_0 e un parametro di dispersione a , si indica con $DP(D_0, a)$ e consiste proprio in una distribuzione di distribuzioni. Un campione $G \sim DP(D_0, a)$ del DP è una distribuzione dei feature vector nei cluster in una singola iterazione del CRP; i campioni presi da G sono gli assegnamenti effettuati per ciascun vettore in un dato cluster.

Ricapitolando, il CRP crea un possibile set di assegnamenti a cluster per i dati che riceve in input; eseguendolo per un determinato numero di volte, si costruisce un processo di Dirichlet, ovvero una distribuzione di distribuzioni, dove tali distribuzioni descrivono gli assegnamenti a cluster di una data esecuzione del CRP. Quello che si ottiene, i termini numerici, sono le probabilità che un nuovo dato (un nuovo feature vector) venga assegnato ad un cluster piuttosto che ad un altro. A questo punto, serve un metodo che permetta di ottenere un *buon* set di assegnamenti. Il funzionamento del *campionamento di Gibbs*, ovvero la procedura che consente di ottenere un buon set di assegnamenti, viene descritto in modo approssimato nel seguito.

- I feature vector vengono assegnati ai cluster in maniera casuale;
- A questo punto, gli assegnamenti effettuati vengono salvati. L'identificatore di feature vector viene selezionato e viene assegnato ad un cluster esistente oppure ad uno nuovo con una probabilità campionata da un'iterazione del CRP che dipende dagli assegnamenti e dai valori di tutti gli altri identificatori;

- Gli step precedenti vengono ripetuti finché non si ottiene un buon set di assegnamenti.

In figura 4.3 vi è un possibile esito del DPMM applicato sui feature vector di ciascun sensore. Il CRP non è l'unico modo per rappresentare l'assegnazione dei dati ai cluster; vi sono altri modelli, come il *Polya-Urn Model* o lo *Stick-Breaking Process* descritti in [13] che, tuttavia, non vengono approfonditi in questo lavoro, poiché l'implementazione utilizzata, disponibile in [12], modella l'assegnazione stessa con il CRP. Un'ultima precisazione riguarda le due varianti del DPMM nominate all'inizio della sezione. La variante gaussiana, detta GDPMM, viene utilizzata quando vanno raggruppati dati continui, mentre la variante multinomiale, detta MDPMM, viene sfruttata quando vengono analizzati documenti testuali; nel software, dunque, è possibile testare l'efficacia della seconda su dati continui, mentre la scelta preferita rimane la prima.

4.4 Hierarchical Dirichlet Process

Una volta terminate le esecuzioni multiple del DPMM, vengono costruite le parole artificiali e successivamente i documenti, come descritto nella sezione 4.2; viene precisato, inoltre, che due cluster diversi individuati durante la prima fase possono fare riferimento allo stesso mezzo di trasporto utilizzato in momenti diversi descritti all'interno dello stesso documento. Per poter catturare quest'informazione, dunque, viene sfruttato l'HDP, ovvero lo *Hierarchical Dirichlet Process*.

L'HDP è un algoritmo di clustering non parametrico bayesiano sviluppato per raggruppare dati già precedentemente raggruppati. Ciò che viene fatto, dunque, è clustering di cluster. L'idea generale che sottende il clustering di dati raggruppati è quella di ottenere informazioni utili a partire da un insieme di documenti generati da un determinato numero di topic¹; tali topic, in questo caso, rappresentano i diversi mezzi di trasporto. Il funzionamento di esso, descritto in [40], si basa su un'estensione del CRP, utilizzato per descrivere il DPMM, chiamata *Chinese Restaurant Franchise*. Ogni cluster da raggruppare è descritto da un processo di Dirichlet, che a sua volta si basa sul CRP per effettuare l'assegnazione dei dati di partenza ad un determinato gruppo.

Si supponga, dunque, di avere J cluster di dati, dove il numero degli stessi per ciascuno dei cluster è n_j (x_{ji}, \dots, x_{jn_j}). Essi vengono presi in considerazione

¹Non è un caso se tali termini indicano proprio i concetti utilizzati per descrivere l'input e l'output di questa fase del lavoro.

in base al loro mixture model, ovvero secondo una descrizione delle caratteristiche dei dati contenuti all'interno di essi, dove i dati sono stati assegnati sfruttando il CRP. Il punto chiave è individuare quei gruppi diversi che, tuttavia, possiedono un mixture model simile, anche se i dati contenuti al loro interno, chiamati anche mixture component, sono in proporzioni diverse. I gruppi che condividono questa caratteristica possono essere generalizzati in un unico nuovo gruppo.

L'analogia che illustra il CRF, dunque, è la seguente. Ci sono J ristoranti, dove in ciascuno di esso sono presenti n_j clienti; ogni cliente si siede ad un tavolo del ristorante in cui si trova. Ad ogni tavolo viene servita una portata tratta da un menù comune a tutti i ristoranti del franchise. I clienti del franchise stesso tendono ad essere socievoli, dunque preferiscono i tavoli più popolati e le portate più popolari; la portata che viene servita ad un insieme di tavoli provenienti da diversi ristoranti del franchise rappresenta il raggruppamento di cluster desiderato. Si hanno, dunque, tante esecuzioni dell'algoritmo quanti sono i documenti forniti in input, perciò i nuovi gruppi vengono creati localmente ad ognuno dei documenti stessi. A questo punto, è necessario un metodo per poter confrontare le topic mixture ottenute; si sale idealmente di uno scalino nel livello di granularità delle operazioni, per individuare a livello globale l'insieme dei documenti nel quale è stato utilizzato lo stesso mezzo di trasporto. Anche per l'HDP esistono altri modelli per il suo funzionamento, come lo *Hierarchical Poly-Urn Scheme*, trattati in [40] ma non in questo lavoro perché l'implementazione utilizzata sfrutta l'analogia alla base del CRP, come per il DPMM, espansa nel CRF.

4.5 Affinity Propagation

Le topic mixture ottenute per ciascun documento vengono fornite in input all'ultimo algoritmo di clustering utilizzato, chiamato AP, ovvero *Affinity Propagation*, descritto approfonditamente in [6] e [41]. Si ricorda che una topic mixture è l'insieme dei topic che va a formare il documento stesso dove, in questo caso, con topic si intende mezzo di trasporto. L'idea chiave consiste nel raggruppare le topic mixture ottenute dai documenti stessi in base alla loro similarità; intuitivamente, se esse sono simili, i documenti a cui fanno riferimento vengono inseriti nello stesso cluster; ciascuno di essi conterrà tutti i punti dove un determinato mezzo di trasporto è stato utilizzato durante la sessione di registrazione.

Affinity Propagation, dunque, è un algoritmo di clustering basato sul concetto dello "scambio di messaggi" tra i dati da raggruppare al fine di trovare un "esemplare" che rappresenti ciascun cluster individuato. Tale algoritmo può

essere inserito nella categoria dei *k-medoids*. Tale termine fa riferimento a una tecnica classica che raggruppa il dataset di partenza costituito da n istanze in k cluster conosciuti a priori. In particolare, in questi metodi il cosiddetto esemplare è il *centroide*² di ciascun cluster e i dati rimanenti vengono raggruppati lavorando con matrici di distanze dai vari centroidi (esemplari) individuati. La caratteristica che distingue Affinity Propagation da altre tecniche k-medoids consiste nel fatto che non è necessario indicare a priori il numero k di cluster da trovare.

Inizialmente, AP considera tutti i dati da raggruppare, simultaneamente, come potenziali esemplari. Ogni dato viene visto come nodo in un grafo e da ciò è stato derivato un metodo che, ricorsivamente, trasmette messaggi a valori reali ai vertici del grafo stesso finché un buon set di esemplari non emerge. I messaggi trasmessi, dunque, vanno a determinare quale dato sarà l'esemplare finale a discapito di altri. In ogni momento, la magnitudo di ciascun messaggio indica l'affinità con cui un dato ne indica un altro come suo esemplare; per tale motivo, l'algoritmo si chiama Affinity Propagation.

Per fare quanto descritto nel paragrafo precedente, AP riceve in input, oltre al dataset da raggruppare, anche una funzione di similarità mediante la quale costruire la corrispondente matrice; tale matrice è quadrata (di dimensione $n \times n$, dove n è il numero di dati contenuti nel dataset) e simmetrica, dove si ha che $m[i][j] = m[j][i]$. Il valore contenuto in $m[i][j]$, ad esempio, indica quanto il dato di indice i è adatto come esemplare per il dato di indice j . Ciascun valore di similarità tra dati è calcolato con la seguente formula:

$$similarity(x_i, x_j) = -||x_i - x_j||^2 \quad \text{con } x_i \text{ e } x_j \text{ dati del dataset.} \quad (4.1)$$

Quella che si ottiene, dunque, è una matrice di distanze euclidee negative, come si può osservare nell'esempio in figura 4.16. Poiché AP non richiede di specificare in anticipo il numero di cluster da individuare, tale valore viene definito durante la fase di scambio dei messaggi.

Ci sono due tipologie di messaggi che possono essere scambiati, ed ognuna di esse codifica una forma di competizione tra i dati. Lo scambio può essere interrotto in qualsiasi momento e le due tipologie possono essere combinate assieme per stabilire quali dati sono gli esemplari cercati e, per tutti gli altri, a quale esemplare fanno riferimento. La *responsabilità* $r[i][j]$ inviata dal dato i al candidato esemplare j indica quanto j è adeguato ad essere un esemplare per i , prendendo in considerazione tutti gli altri potenziali candidati per il dato i . La *disponibilità* $d[i][j]$ inviata dal candidato esemplare j al dato i

²Consiste nella "posizione media" di tutti i suoi punti, ovvero la media aritmetica delle posizioni di ciascuno di essi.

indica quanto risulta essere appropriato per i scegliere proprio j come proprio esemplare, prendendo in considerazione anche il supporto proveniente da altri dati affinché lo stesso j sia scelto in tal senso. Si hanno, dunque, due ulteriori matrici di dimensione pari a quella della similarità e lo scambio di messaggi consente nell'aggiornare i valori nelle matrici di responsabilità e disponibilità (d) coinvolgendo quelli di similarità (s). La responsabilità tra due dati (r), infatti, viene definita secondo la seguente formula, posto $j \neq k$:

$$r(x_i, x_j) = s(x_i, x_j) - \max(d(x_i, x_k) - s(x_i, x_k)) \quad (4.2)$$

Per la formula della disponibilità si approfondisca in [41] o [6]. Sfruttando i meccanismi basati su queste analogie che portano all'aggiornamento competitivo e contemporaneo di tre matrici i cluster si formano iterazione dopo iterazione. In qualsiasi momento si può fermare la propagazione dell'affinità e i valori di responsabilità e disponibilità possono essere combinati per identificare gli esemplari. Solitamente si sceglie di interrompere il processo dopo un ragionevole numero di iterazioni; nell'implementazione utilizzata in questo lavoro il numero scelto è duecentotrenta, valore predefinito impostato dall'autore dell'algoritmo. Come descritto nella sezione 4.2, l'input di questa fase sono le topic mixture dei documenti e una volta terminata si ottengono i cluster formati dagli stessi documenti, dove ognuno dei cluster forma un mezzo di trasporto, ottenuti sfruttando le tre matrici descritte in questa sezione; la matrice di similarità, in particolare, viene costruita basandosi sulle topic mixture stesse. In figura 4.17 è possibile osservare una rappresentazione dello scambio di messaggi effettuato dall'algoritmo, mentre nella 4.18 si può osservare una schematizzazione generale del suo funzionamento.

4.6 Analisi risultati

Scopo di questa sezione è presentare i risultati ottenuti ed eventuali considerazioni relative ai test effettuati eseguendo il processo di clustering sull'intera settimana di dati utilizzati nella sezione 3.9 con l'aggiunta di una giornata extra e su sottoinsiemi di essa provenienti dall'applicazione Lifetracker, o su ulteriori insiemi di dati provenienti dai dataset pubblici, indicati nella tabella 3.2. Per ciascuna tipologia di dataset vengono descritte le caratteristiche di ciascun insieme testato, le attività presenti all'interno di essi e la precisione finale del processo di clustering relativa all'insieme stesso. Nel paragrafo seguente si intende riportare alcune considerazioni riguardo ai risultati disponibili nella tabella 4.1, mentre in quello successivo vengono analizzate ulteriori problematiche individuate.

I dataset indicati nella tabella 3.2 vengono testati con un feature set appropriato, dove con tale aggettivo si intendono i set più efficaci per i sensori disponibili per la tipologia del dataset analizzato; lo scopo è verificare il comportamento dei classificatori migliori e dell'approccio non parametrico sviluppato. Per quanto riguarda Geolife, prima tipologia di dataset testata, costituita dai dati del solo GPS ([42]), quello che si può notare è che la composizione delle diverse settimane analizzate influenzi le performance di entrambe le tecniche. Tale variabilità nei risultati è dovuta proprio alla sola presenza del GPS, che risulta insufficiente per svolgere il riconoscimento in modo efficace. La seconda tipologia di dati testati, Twente, è caratterizzata dai dati di accelerometro, giroscopio e magnetometro ([43]). Osservando il risultato dell'analisi di TW4 si può notare come il magnetometro, da solo, spicchi sui sensori rimanenti. Ciò è dovuto, con una buona probabilità, alle feature calcolate per esso in questo lavoro e va a confutare quanto detto dagli autori di tale tipologia di dataset in [44]. Osservando i risultati per TW5 si può notare come la combinazione dei tre sensori fornisca le prestazioni migliori, a differenza dell'analisi di ciascuno di essi svolta singolarmente o a coppie, sia per i classificatori che per il processo di clustering, confermando ulteriormente l'ipotesi illustrata nella sezione 3.9. Per quanto riguarda la tipologia in-house di dataset, prodotta mediante Lifetracker, si osservino i risultati di IH2, analizzato inizialmente con il solo GPS e, successivamente, aggiungendo accelerometro, giroscopio ed infine magnetometro, ossia con i feature set quattro, cinque, sette ed otto; anche per il processo di clustering si conferma che l'aggiunta di tali sensori migliora le prestazioni finali. Osservando i rimanenti dataset IH_i testati, è che la composizione delle singole giornate da analizzare influenza di molto i risultati del processo, mentre i classificatori subiscono tale fenomeno in misura molto minore. Inoltre, si può vedere come i risultati finali dei classificatori stessi aumentino dove quelli del processo, a loro volta, aumentano e viceversa; nella maggior parte dei casi, dunque, tali valori sono allineati. Va notato, inoltre, come le performance di J48 e RandomForest siano stabilmente superiori alla soglia del novanta per cento quando il dataset da analizzare viene riassunto con l'ottavo feature set.

Una problematica comune a qualsiasi insieme di dati analizzato indipendentemente dal dataset di provenienza è costituita dalla dimensione della sliding window e dal fattore di sovrapposizione. Tali parametri dipendono dal numero di dati da raggruppare; infatti, se sono inferiori rispetto ad una certa soglia, uno degli algoritmi facenti parte del processo di clustering, l'HPD, fallisce nella maggior parte dei casi. Quello che va fatto, dunque, è assegnare i valori ai due parametri in modo euristico per ogni insieme di dati analizzato facendo molta attenzione. Il rischio è di ottenere risultati falsi che non rispecchiano la reale

bontà del processo. La prima tipologia di dataset testata è Geolife. I valori di precisione che si ottengono nei vari insiemi analizzati non sono del tutto soddisfacenti ma nemmeno fallimentari, nella maggior parte dei casi. Le motivazioni vanno ricercate proprio nella sola presenza di dati provenienti dal GPS; analizzando i cluster di mezzi di trasporto che si formano alla fine del processo si può notare come la sola presenza di questo sensore non è in grado di individuare quei mezzi che occorrono di meno nell'insieme analizzato; tendono a comparirne uno o due predominanti rispetto agli altri e solamente essi vengono assegnati correttamente ai dati grezzi corrispondenti. In altri casi, inoltre, nonostante ve ne siano effettivamente due predominanti, ne viene individuato solamente uno, danneggiando ulteriormente le prestazioni. La precisione finale, dunque, è caratterizzata solamente da quelle istanze dell'insieme analizzato che descrivono proprio i mezzi predominanti. Quando si hanno tre o più attività distribuite più o meno uniformemente nel dataset le prestazioni crollano; tale fenomeno nel dataset prodotto mediante Lifetracker non avviene. Per quanto riguarda il secondo dataset pubblico, proveniente dall'università di Twente, i valori di precisione che si ottengono, sono nella maggior parte dei casi peggiori di quelli dei dati Geolife. Tale aspetto conferma che il GPS rimane il sensore più adatto per soddisfare l'obiettivo di questo lavoro e che i rimanenti tre sensori non sono in grado di fornire prestazioni adeguate, se analizzati separatamente. Anche in un approccio non parametrico, dunque, è la combinazione di tutti e quattro i sensori stessi a fornire i risultati migliori, con il GPS che fa il grosso del lavoro mentre quelli rimanenti si limitano ad alzare leggermente l'asticella della precisione finale. Un'ulteriore caratteristica negativa di questo dataset è che i dati dei sensori vengono registrati per cinquanta volte al secondo; è improbabile che vi siano sostanziali differenze nei dati grezzi in una finestra temporale così ridotta. Il fatto che vi sia una tale densità di dati molto simili tra di loro in così poco tempo, dunque, può incidere negativamente sulle prestazioni del processo di clustering; non è importante, dunque, solamente quali sensori vengono analizzati, ma anche la frequenza con la quale i loro dati vengono campionati. Un campione al secondo è più che sufficiente per raggiungere l'obiettivo posto in partenza, senza portare gli algoritmi di clustering all'interno del processo a trarre conclusioni errate. Per confermare quanto detto in questo paragrafo, nella tabella precedentemente citata sono presenti i risultati del processo stesso eseguito su ogni sensore preso singolarmente. L'ultimo dataset analizzato è quello in-house, prodotto tramite Lifetracker, con un'adeguata frequenza di campionamento. Un ulteriore commento relativo al processo di clustering è che riesce ad individuare diversi "stili" d'uso di ciascun mezzo, come indicato nella sezione 4.2; per esempio, in un dataset potrebbero comparire tre cluster di "car"; solitamente ciò avviene perché i dati di uno o più

Classificatore				Clustering	
Tipologia		Alberi		Mixture Model	
Feature Set	Dataset	J48	Ran.Forest	DPGMM+HDP+AP	Migliore
3	GL1	70,16%	64,74%	42,50%	J48
3	GL2	91,80%	91,24%	63,70%	J48
3	GL3	77,53%	73,89%	80,69%	DPGMM+HDP+AP
11	TW1	47,91%	41,30%	25,03%	J48
12	TW2	50,82%	44,51%	21,49%	J48
13	TW3	61,38%	53,68%	21,43%	J48
14	TW4	54,92%	52,02%	27,49%	J48
15	TW5	68,20%	70,04%	29,90%	Ran.Forest
8	IH1	91,36%	91,08%	60,43%	J48
4	IH2	88,26%	88,75%	55,41%	Ran.Forest
5	IH2	89,88%	89,14%	59,40%	J48
7	IH2	89,12%	89,56%	64,17%	Ran.Forest
8	IH2	90,43%	92,05%	85,77%	Ran.Forest
8	IH3	92,34%	91,47%	58,13%	J48
8	IH4	93,07%	94,55%	54,50%	Ran.Forest
8	IH5	95,87%	96,56%	83,30%	Ran.Forest
8	IH6	96,50%	96,37%	69,90%	J48
8	IH7	93,65%	94,16%	63,71%	Ran.Forest
8	IH8	95,42%	96,48%	72,16%	Ran.Forest

Tabella 4.1: Performance del processo di clustering per ciascun dataset analizzato.

sensori riassunti nei feature vector sono significativamente diversi tra i cluster; in definitiva, il numero di quelli individuati è maggiore o uguale al numero dei mezzi. Tale caratteristica può essere analizzata osservando le composizioni dei cluster stessi nel rapporto finale del processo, ottenibile mediante un pulsante dell'interfaccia in figura 4.15, come quello in figura 4.10.

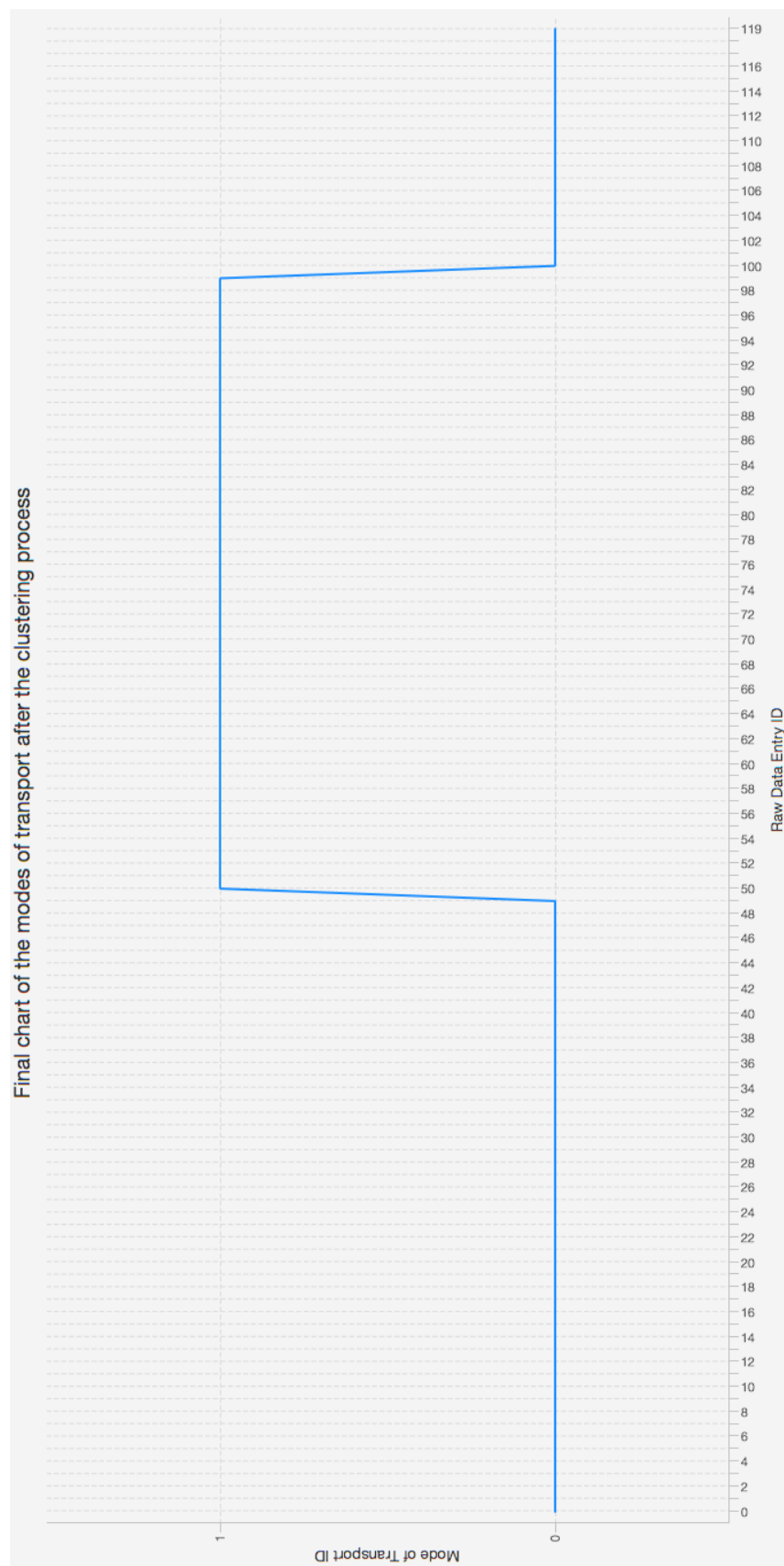


Figura 4.12: Esempio di grafico con gli identificatori dei mezzi di trasporto individuati dal processo di clustering per ciascun dato grezzo.

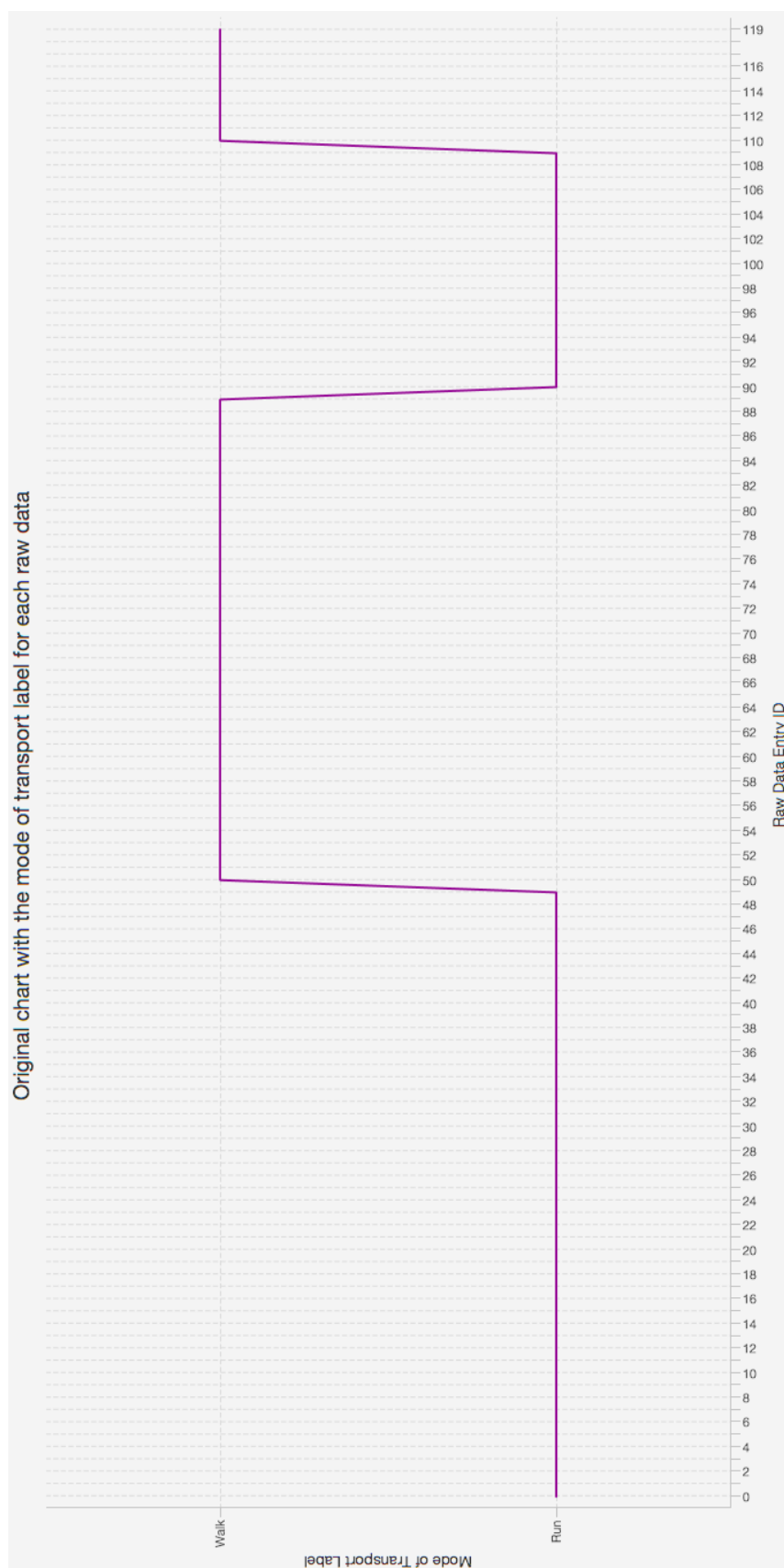


Figura 4.13: Esempio di grafico contenente l'evoluzione temporale dei mezzi di trasporto veramente utilizzati per ciascun dato grezzo.

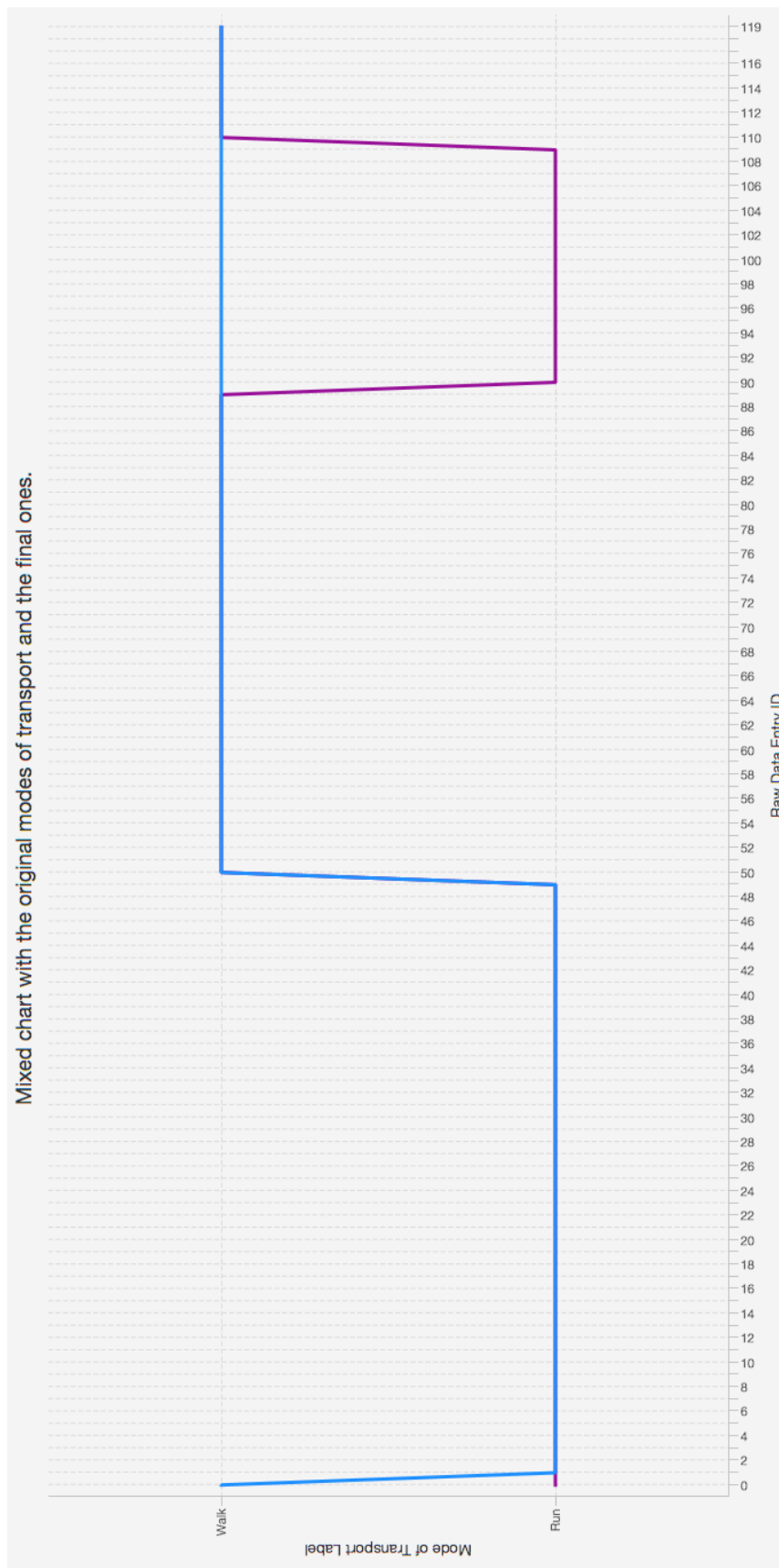


Figura 4.14: Esempio di grafico costruito per confrontare quanto dedotto dal processo di clustering con quanto è successo nella realtà.

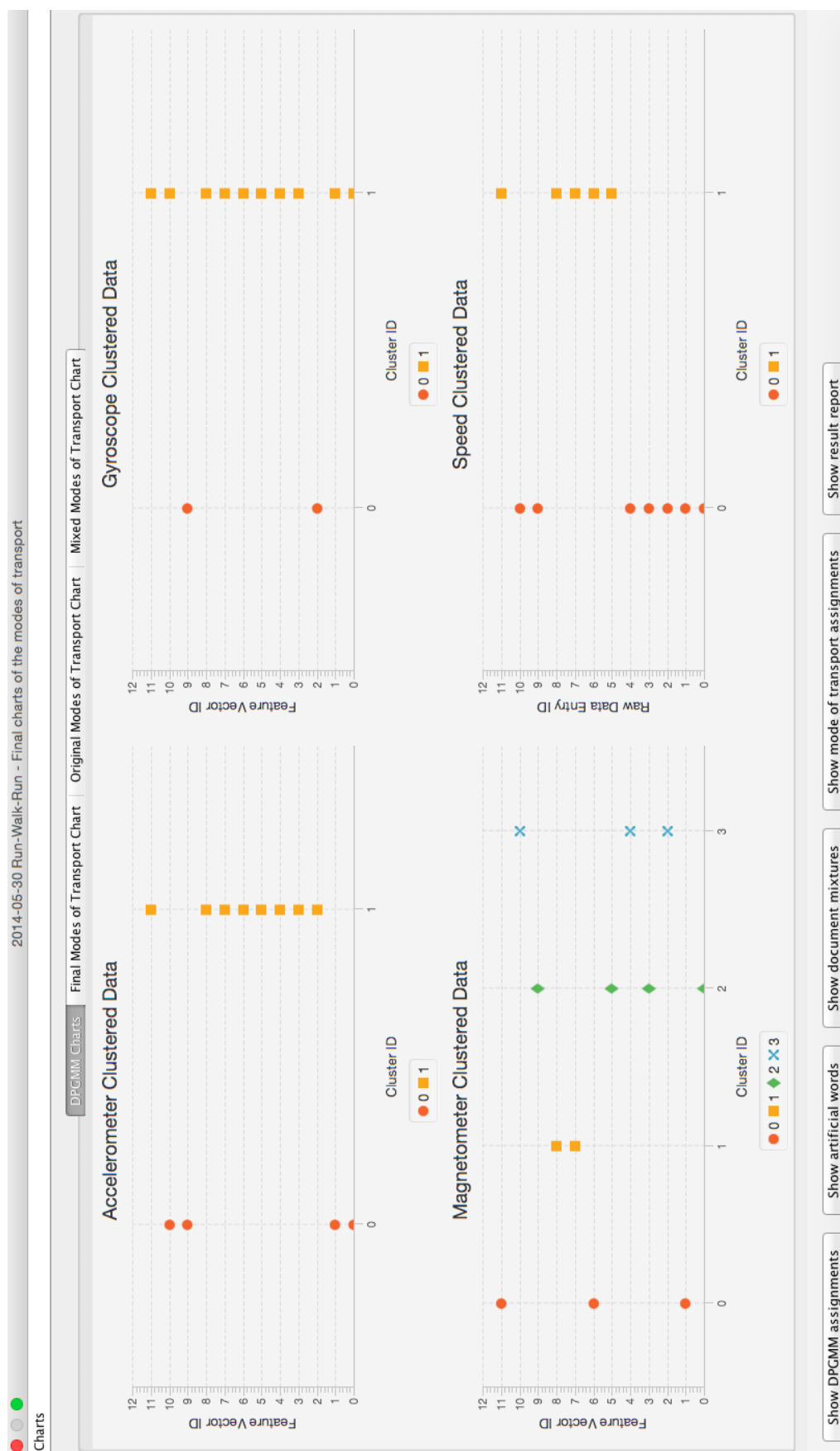


Figura 4.15: Terza finestra dell'interfaccia grafica del software desktop ed esempi di grafici relativi agli assegnamenti delle esecuzioni del DPMM.

```

0.0 -0.00500000000000000044 -0.035000000000000001
-0.00500000000000000044 0.0 -0.019999999999999997
-0.035000000000000001 -0.019999999999999997 0.0

```

Figura 4.16: Esempio di matrice di similarità costruita da Affinity Propagation.

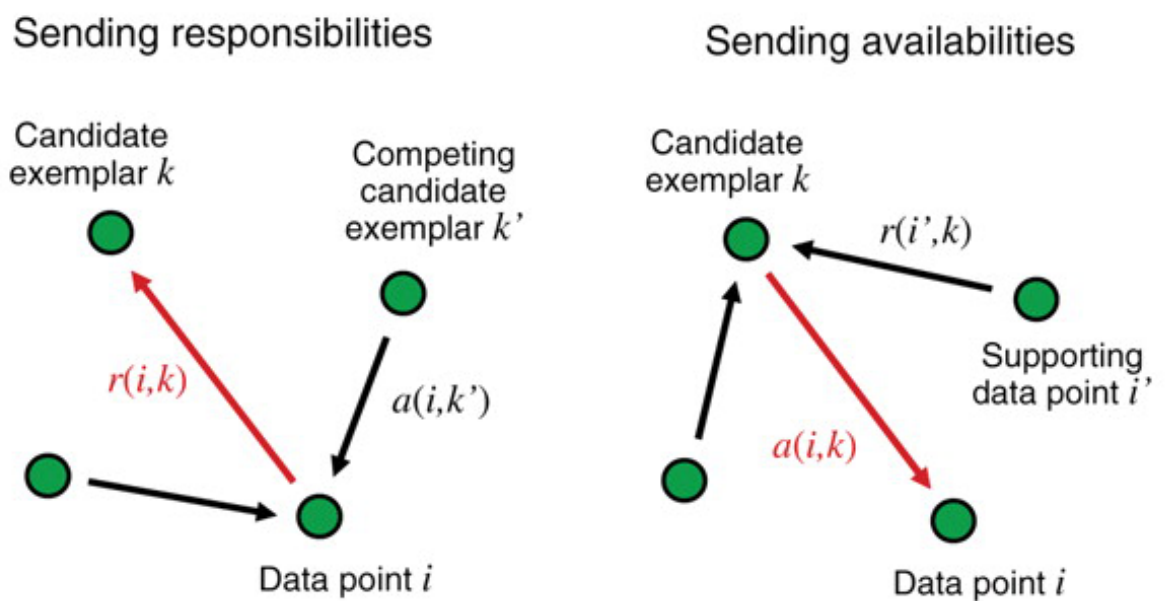


Figura 4.17: Scambio di messaggi nell’Affinity Propagation - (Fonte: Science Magazine)

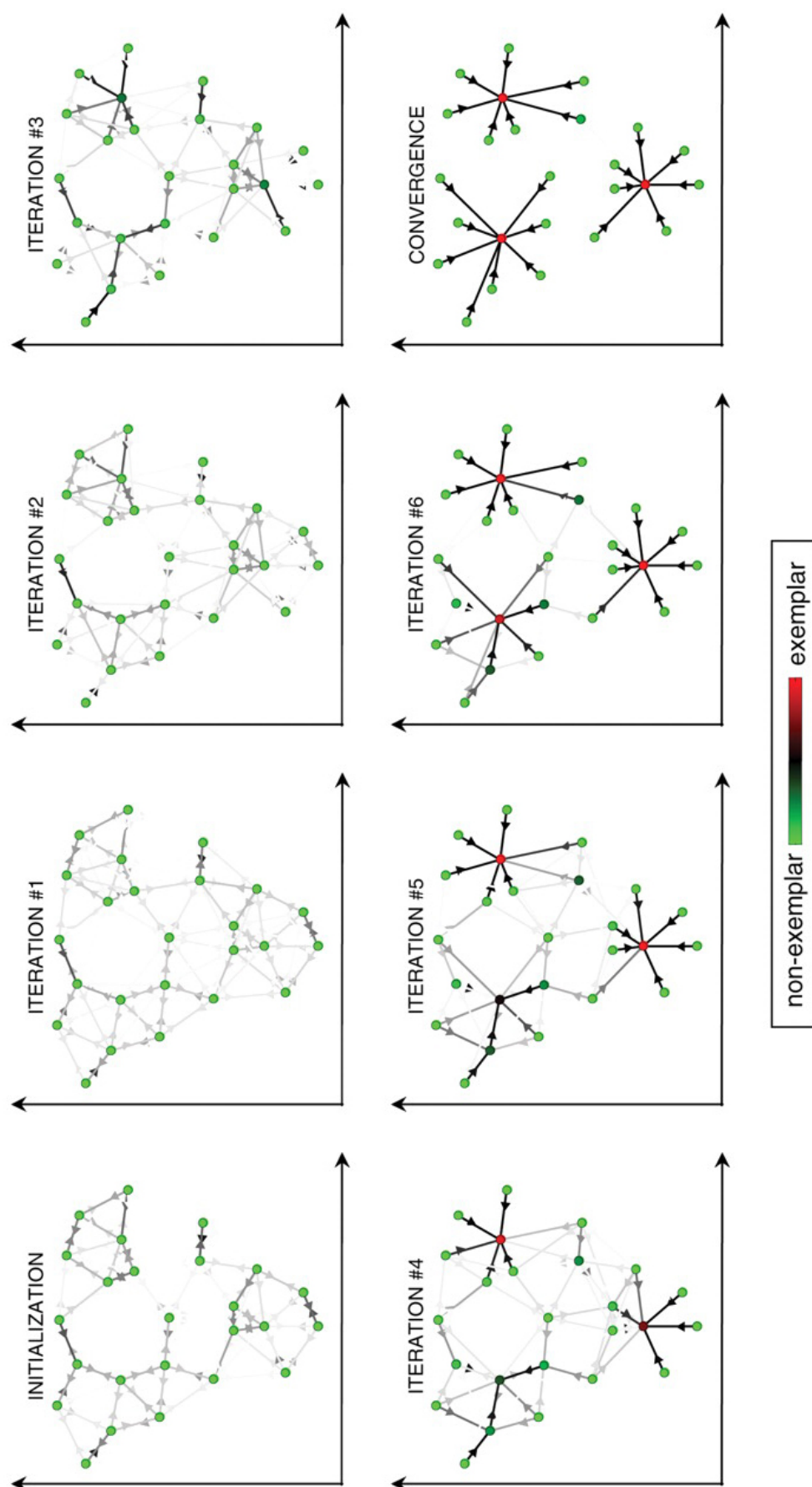


Figura 4.18: Schema descrivente il funzionamento di Affinity Propagation - (Fonte: Science Magazine)

Capitolo 5

Conclusioni

In questo lavoro si è dimostrato come sia possibile riconoscere il mezzo di trasporto utilizzato dagli utenti di un'applicazione installata su un dispositivo mobile tramite dati grezzi registrati da sensori quali accelerometro, giroscopio, magnetometro e GPS, contenuti all'interno del dispositivo stesso, in una determinata finestra temporale. Ciò viene svolto tramite un software in grado di applicare due grandi tecniche di machine learning, con il fine ultimo di ottenere un approccio non parametrico al riconoscimento. Inizialmente, vengono descritte le operazioni di elaborazione e riduzione di dimensione e dimensionalità dei dati forniti in input, effettuate mediante il calcolo dei feature vector (dove le feature presenti in ciascuno di essi vengono decise dall'utente), necessarie prima di procedere con l'analisi vera e propria. La prima grande tecnica di machine learning utilizzata in questo lavoro è la classificazione, ed è caratterizzata da un approccio supervisionato alla risoluzione del problema. Sono state testate diverse tipologie di classificatore mediante un software esterno chiamato Weka, sfruttando diversi set di feature, fino ad individuare quello in grado di fornire le prestazioni migliori per il dataset in oggetto con il set di feature più adatto. Successivamente è stata analizzata la seconda delle tecniche di machine learning utilizzate, il clustering, caratterizzata da un approccio non supervisionato; è stato implementato un processo coinvolgente tre diversi algoritmi appartenenti a tale tecnica ed alcune operazioni intermedie per ottenere una versione non supervisionata e non parametrica del riconoscimento dei mezzi di trasporto. Per ciascuna delle due tecniche sono stati evidenziati pregi, difetti e problematiche in apposite sezioni di analisi dei risultati, dove il tutto viene corredato da immagini, dati ed esempi. Clustering e classificazione, infine, vengono testate anche su dataset di dominio pubblico derivanti dal progetto Geolife di Microsoft Research (costituito solamente dai dati del GPS) e dal lavoro di ricerca fatto dall'università olandese della città di Twente (costituito dai dati di accelerometro, magnetometro e giroscopio), per validare le

ipotesi fatte nel corso del lavoro sfruttando il dataset prodotto autonomamente mediante l'applicazione Lifetracker (costituito da tutti e quattro i sensori citati). Analizzando il primo dei due dataset pubblici si prova che i dati del solo GPS non consentono di effettuare il riconoscimento con un buon livello di precisione. Inoltre, si vede come vengano effettivamente riconosciuti solamente il mezzo più utilizzato o, al più, i primi due; quelli che occorrono di meno nel dataset non vengono mai presi in considerazione. In generale, all'aumentare del numero dei mezzi stessi le prestazioni crollano. Analizzando il secondo dataset pubblico, si scopre che i tre sensori di cui si compone, presi singolarmente, forniscono prestazioni addirittura peggiori del solo GPS. Se considerati assieme, invece, le prestazioni rimangono peggiori rispetto al primo dataset pubblico, tuttavia risultano essere migliori rispetto all'analisi singola dei tre sensori. L'ipotesi che viene fatta analizzando il dataset prodotto con Lifetracker, supportato anche dai risultati dei classificatori, è che l'analisi combinata di tutti e quattro i sensori, con tutti i set di feature, sia la soluzione migliore per raggiungere l'obiettivo. Tale assunzione si rivela corretta ed il riconoscimento viene effettuato in modo non parametrico, con buone percentuali di successo, indipendentemente dal numero di mezzi; anche quelli che occorrono di meno vengono assegnati correttamente, in una buona parte dei casi. In ultima analisi, dunque, l'obiettivo posto nella sezione 1.2 si può considerare raggiunto; tuttavia, il percorso per poter realizzare un prodotto effettivamente utilizzabile da una platea di utenti inesperti, tuttavia, non si esaurisce; sono necessari ulteriori passi in avanti, delineati nella sezione 5.1.

5.1 Sviluppi futuri

Nel seguito si intende presentare alcune importanti questioni da approfondire per poter giungere alla creazione di un'applicazione mobile effettivamente utilizzabile da utenti inesperti. Tali questioni sono:

- *Stima automatica della dimensione della finestra scorrevole*: nella sezione 4.2 viene spiegato come tale parametro, unito al fattore di sovrapposizione, dipende dal numero effettivo di feature vector da raggruppare. Nel software implementato per le analisi tali valori vengono inseriti manualmente ma sarebbe auspicabile, in futuro, avere una funzione in grado di calcolare automaticamente questo parametro.
- *Stima automatica della dimensione dell'intervallo temporale per il calcolo di ciascun feature vector*: questo parametro viene influenzato dal tempo totale di registrazione dei dati grezzi per il dataset fornito in input. Anche in questo caso, dunque, sarebbe utile avere una funzione in grado

di calcolare tale intervallo temporale, nonostante si sia notato che un intervallo di dieci secondi risulta essere un buon compromesso per ogni tipologia di dataset.

- *Modulo di analisi dei dati contenuti in ciascun cluster*: gli assegnamenti finali delle etichette ai dati grezzi vengono svolti secondo un'approssimazione; poiché i dati sono etichettati, infatti, una volta raggruppati in cluster, il mezzo di trasporto che viene assegnato è quello la cui etichetta compare più volte all'interno del cluster stesso. Inevitabilmente si tratta di un'assunzione che porta ad errori ma che, tuttavia, risulta essere sufficiente ai fini del testing del software. In una situazione d'uso reale non si ha a disposizione l'informazione fornita dall'etichettatura dei dati, perciò è necessario sviluppare un modulo all'interno del software che, una volta ricevuti in input i cluster individuati dal processo, vada ad eseguire l'etichettatura di ciascun dato grezzo analizzando le caratteristiche degli attributi dei feature vector contenuti all'interno dei cluster stessi.
- *Sviluppo di una rinnovata applicazione mobile con relativa componente lato server*: una volta risolte le due questioni precedenti, nel cuore del software, si può procedere verso la realizzazione del prodotto finale sviluppando un sistema distribuito caratterizzato da una componente lato server contenente tutta la logica dell'applicazione (poiché il software è scritto in Java, può essere facilmente trasformato in tale componente lato server sfruttando la tecnologia JSP¹) e da una nuova versione dell'applicazione mobile che dovrà essere in grado di registrare i dati dei sensori, come già avviene, per poi inviarli al server. A questo punto, essa dovrà attendere l'elaborazione degli stessi per poi occuparsi della visualizzazione dei risultati ottenuti a beneficio degli utenti.
- *Calcolo di un ulteriore set di feature*: per allinearsi a quanto spiegato in [3], articolo dal quale è stata derivata l'ossatura del processo di clustering implementato in questo lavoro, con il fine ultimo di effettuare ulteriori valutazioni e verifiche della bontà degli algoritmi implementati, è necessario il calcolo di un nuovo set di feature per il GPS, costituito dai valori di *heading direction change rate*, che corrisponde al rapporto tra il numero di volte in cui la misura della variazione di direzione presente tra due distinte rilevazioni GPS supera una determinata soglia e la distanza

¹JavaServer Pages, di solito indicato proprio con la sigla JSP è una tecnologia di programmazione lato server in Java per lo sviluppo della logica ed eventualmente della presentazione (tipicamente secondo il pattern MVC) di applicazioni Web, fornendo contenuti dinamici in formato HTML o XML.

percorsa tra di esse, dello *stop rate*, il rapporto tra il numero di fermate intercorse e, nuovamente, la distanza percorsa ed infine del *velocity change rate*, che corrisponde al rapporto tra il numero di punti in cui la variazione di velocità risulta essere superiore ad una certa soglia e, ancora una volta, la distanza percorsa tra le due rilevazioni analizzate.

5.2 Grafici e dati

Lo scopo di questa sezione è presentare grafici e risultati finali relativi al processo di clustering per alcuni dei set testati ed indicati nella tabella 3.2. Per ogni esempio è presente una breve descrizione del set analizzato, seguita da grafici riportanti la distribuzione dei mezzi di trasporto e l'esito del DPMM. Seguono un grafico che confronta gli assegnamenti originali dei dati grezzi e quelli dopo il processo di clustering stesso e la struttura dei cluster individuati dal processo (si può osservare un esempio minimale in figura 4.10). Infine, è presente un commento per illustrare i risultati ottenuti. Si veda la pagina seguente.

5.2.1 Geolife

Il primo esempio per il dataset derivato dal progetto Geolife è il dataset GL1. Si tratta di una settimana di registrazione di dati GPS, con sei mezzi di trasporto utilizzati.

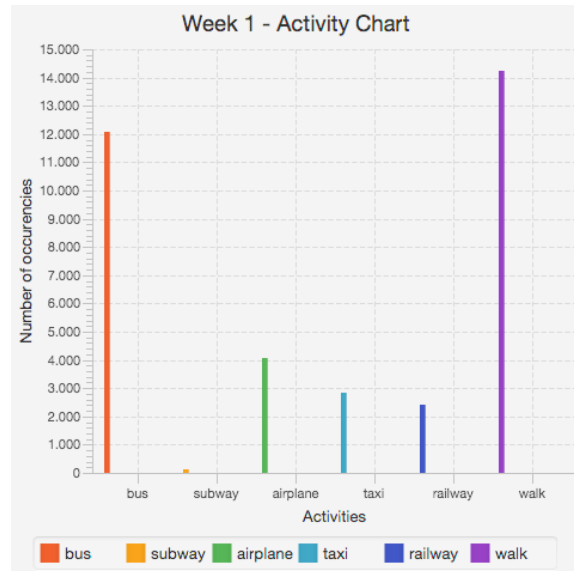


Figura 5.1: Grafico con la distribuzione dei mezzi di trasporto per il dataset GL1.



Figura 5.2: Grafico con l'esito del DPMM per il dataset GL1.

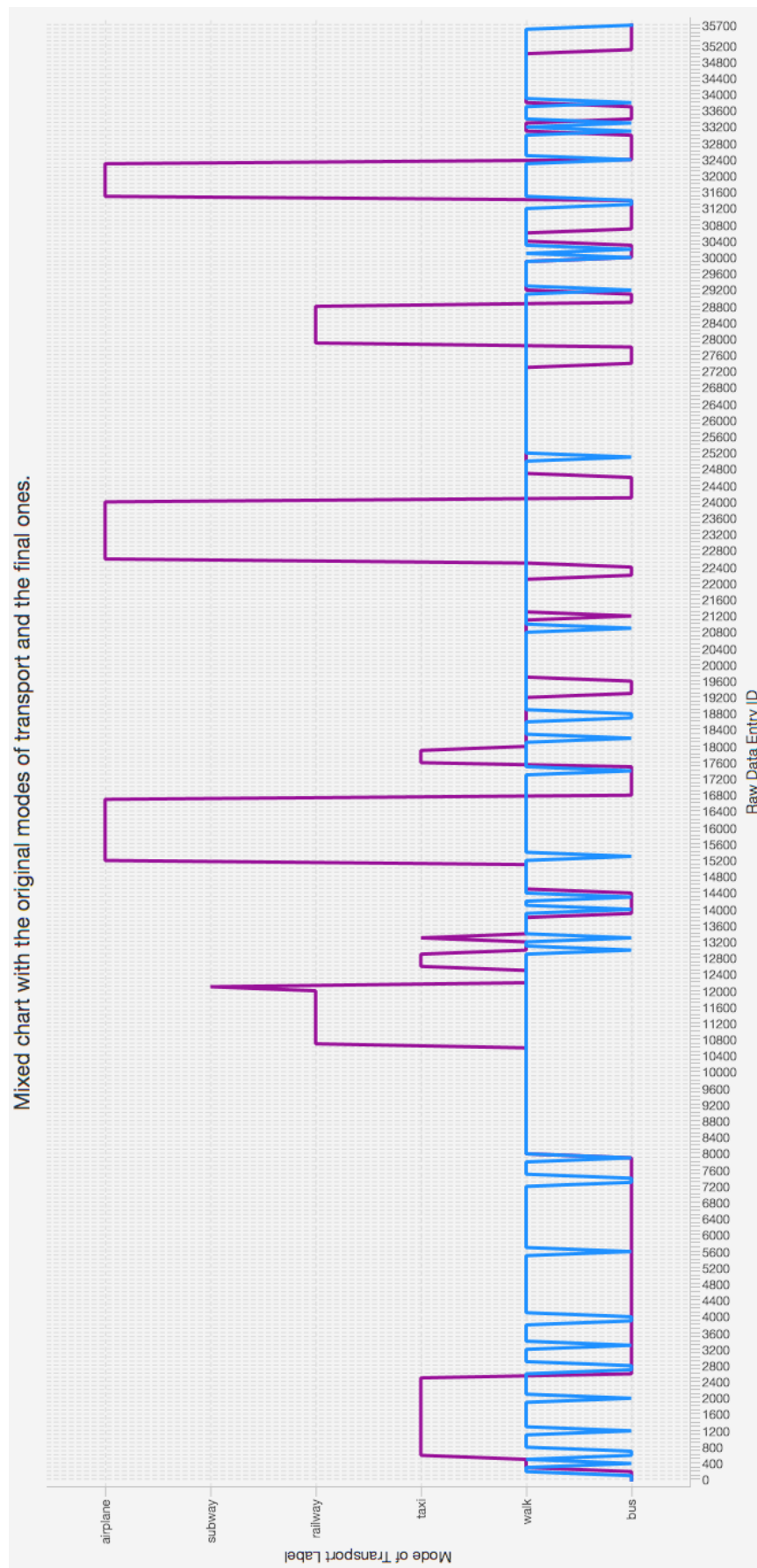


Figura 5.3: Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset GL1.

TOTAL NUMBER OF MODES OF TRANSPORT FOUND: 19

AMOUNT OF SUCCESS: 42.49755%

AMOUNT OF ERROR: 57.50245%

INSTANCES CORRECTLY CLUSTERED: 15178

INSTANCES INCORRECTLY CLUSTERED: 20537

MODE OF TRANSPORT STRUCTURE

MODE OF TRANSPORT ID: 0

LABEL bus – VALUE: 147 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 1

LABEL bus – VALUE: 9771 – PERCENTUAL: 32.00668%

LABEL subway – VALUE: 106 – PERCENTUAL: 0.34722224%

LABEL airplane – VALUE: 3881 – PERCENTUAL: 12.712919%

LABEL taxi – VALUE: 2093 – PERCENTUAL: 6.8560014%

LABEL railway – VALUE: 2337 – PERCENTUAL: 7.6552672%

LABEL walk – VALUE: 12340 – PERCENTUAL: 40.42191%

MODE OF TRANSPORT ID: 2

LABEL bus – VALUE: 70 – PERCENTUAL: 16.666668%

LABEL taxi – VALUE: 110 – PERCENTUAL: 26.190477%

LABEL walk – VALUE: 240 – PERCENTUAL: 57.14286%

MODE OF TRANSPORT ID: 3

LABEL bus – VALUE: 42 – PERCENTUAL: 70.0%

LABEL subway – VALUE: 18 – PERCENTUAL: 30.000002%

MODE OF TRANSPORT ID: 4

LABEL walk – VALUE: 60 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 5

LABEL bus – VALUE: 300 – PERCENTUAL: 50.0%

LABEL walk – VALUE: 300 – PERCENTUAL: 50.0%

MODE OF TRANSPORT ID: 6

LABEL bus – VALUE: 480 – PERCENTUAL: 61.538464%

LABEL taxi – VALUE: 99 – PERCENTUAL: 12.692308%

LABEL walk – VALUE: 201 – PERCENTUAL: 25.76923%

MODE OF TRANSPORT ID: 7

LABEL airplane – VALUE: 60 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 8

LABEL bus – VALUE: 26 – PERCENTUAL: 43.333332%

LABEL walk – VALUE: 34 – PERCENTUAL: 56.666668%

MODE OF TRANSPORT ID: 9

LABEL walk – VALUE: 60 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 10

LABEL bus – VALUE: 60 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 11

LABEL walk – VALUE: 60 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 12

LABEL bus – VALUE: 935 – PERCENTUAL: 38.958332%

LABEL airplane – VALUE: 120 – PERCENTUAL: 5.0%

LABEL taxi – VALUE: 474 – PERCENTUAL: 19.75%

LABEL railway – VALUE: 60 – PERCENTUAL: 2.5%

LABEL walk – VALUE: 811 – PERCENTUAL: 33.791668%

MODE OF TRANSPORT ID: 13

LABEL bus – VALUE: 60 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 14

LABEL taxi – VALUE: 60 – PERCENTUAL: 100.0%

MOST FREQUENT LABEL: taxi – VALUE: 60

MODE OF TRANSPORT ID: 15

LABEL bus – VALUE: 60 – PERCENTUAL: 100.0%

MOST FREQUENT LABEL: bus – VALUE: 60

MODE OF TRANSPORT ID: 16

LABEL bus – VALUE: 60 – PERCENTUAL: 50.0%

LABEL walk – VALUE: 60 – PERCENTUAL: 50.0%

MODE OF TRANSPORT ID: 17

LABEL walk – VALUE: 60 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 18

LABEL bus – VALUE: 60 – PERCENTUAL: 100.0%

Sorgente 5.1: Struttura dei cluster individuati per il dataset GL1

Per quanto riguarda il primo set testato, facendo riferimento all'analisi dei risultati in sezione 4.6, si può notare come la precisione finale sia piuttosto insoddisfacente. Tale risultato si ottiene perché l'analisi dei dati GPS individua solamente il mezzo che occorre il maggior numero di volte nel dataset, in questo caso la camminata, mentre quelli rimanenti non sono quasi mai presi in considerazione dal processo. Il valore di precisione, dunque, è determinato dai soli dati grezzi facenti riferimento proprio alla camminata, che costituiscono, non a caso, il quarantadue per cento del dataset stesso.

La seconda settimana di registrazioni di dati GPS analizzata è il dataset GL2. I mezzi di trasporto contenuti in tale set sono due.

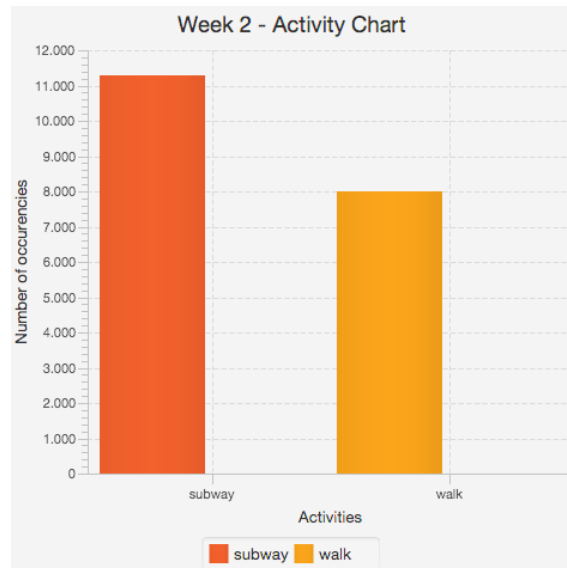


Figura 5.4: Grafico con la distribuzione dei mezzi di trasporto per il dataset GL2.

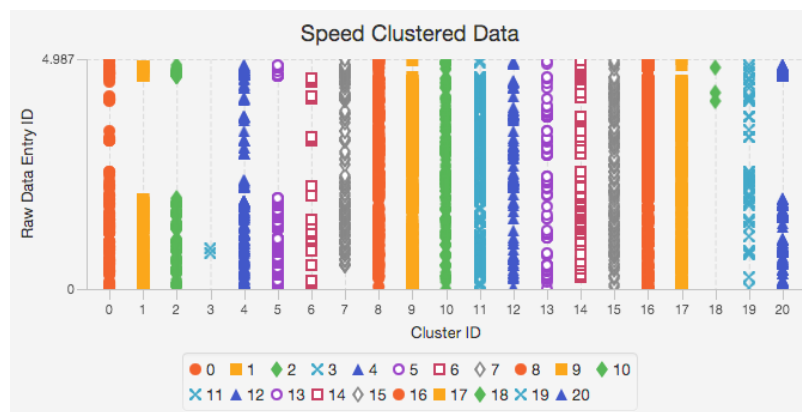


Figura 5.5: Grafico con l'esito del DPMM per il dataset GL2.

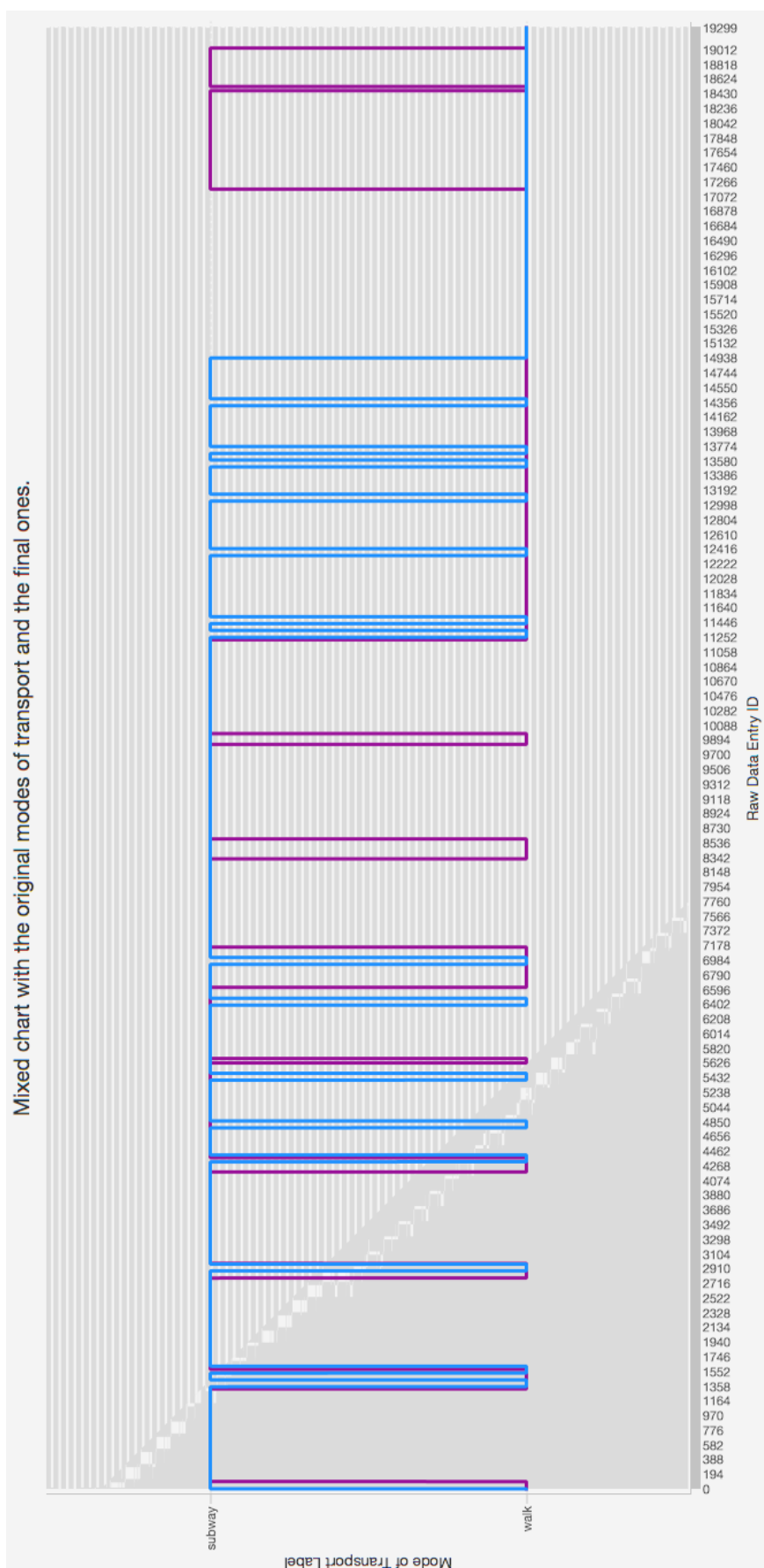


Figura 5.6: Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset GL2.

TOTAL NUMBER OF MODES OF TRANSPORT FOUND: 23

AMOUNT OF SUCCESS: 63.704662%

AMOUNT OF ERROR: 36.295338%

INSTANCES CORRECTLY CLUSTERED: 12295

INSTANCES INCORRECTLY CLUSTERED: 7005

MODE OF TRANSPORT STRUCTURE

MODE OF TRANSPORT ID: 0

LABEL subway – VALUE: 1893 – PERCENTUAL: 42.529766%

LABEL walk – VALUE: 2558 – PERCENTUAL: 57.470234%

MODE OF TRANSPORT ID: 1

LABEL subway – VALUE: 720 – PERCENTUAL: 72.72727%

LABEL walk – VALUE: 270 – PERCENTUAL: 27.272728%

MODE OF TRANSPORT ID: 2

LABEL subway – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 3

LABEL subway – VALUE: 169 – PERCENTUAL: 46.944447%

LABEL walk – VALUE: 191 – PERCENTUAL: 53.055553%

MODE OF TRANSPORT ID: 4

LABEL subway – VALUE: 990 – PERCENTUAL: 55.0%

LABEL walk – VALUE: 810 – PERCENTUAL: 45.0%

MODE OF TRANSPORT ID: 5

LABEL subway – VALUE: 46 – PERCENTUAL: 51.111115%

LABEL walk – VALUE: 44 – PERCENTUAL: 48.88889%

MODE OF TRANSPORT ID: 6

LABEL subway – VALUE: 858 – PERCENTUAL: 56.078434%

LABEL walk – VALUE: 672 – PERCENTUAL: 43.92157%

MODE OF TRANSPORT ID: 7

LABEL subway – VALUE: 5145 – PERCENTUAL: 65.70881%

LABEL walk – VALUE: 2685 – PERCENTUAL: 34.291187%

MODE OF TRANSPORT ID: 8

LABEL walk – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 9

LABEL subway – VALUE: 478 – PERCENTUAL: 66.481224%

LABEL walk – VALUE: 241 – PERCENTUAL: 33.518776%

MODE OF TRANSPORT ID: 10

LABEL subway – VALUE: 25 – PERCENTUAL: 27.777779%

LABEL walk – VALUE: 65 – PERCENTUAL: 72.22222%

MODE OF TRANSPORT ID: 11

LABEL subway – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 12

LABEL walk – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 13

LABEL subway – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 14

LABEL subway – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 15

LABEL subway – VALUE: 90 – PERCENTUAL: 50.0%

LABEL walk – VALUE: 90 – PERCENTUAL: 50.0%

MODE OF TRANSPORT ID: 16

LABEL subway – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 17

LABEL walk – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 18

LABEL subway – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 19

LABEL subway – VALUE: 90 – PERCENTUAL: 50.0%

LABEL walk – VALUE: 90 – PERCENTUAL: 50.0%

MODE OF TRANSPORT ID: 20

LABEL subway – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 21

LABEL subway – VALUE: 90 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 22

LABEL subway – VALUE: 74 – PERCENTUAL: 82.22222%

LABEL walk – VALUE: 16 – PERCENTUAL: 17.777779%

Sorgente 5.2: Struttura dei cluster individuati per il dataset GL2

Rispetto alla prima settimana, le prestazioni del processo di clustering sono migliori; questo perché vi sono solamente due mezzi di trasporto, dove ciascuno di essi è caratterizzato da un elevato numero di dati grezzi. In questo particolare caso, dunque, gli algoritmi riescono ad individuare diversi cluster dove, in alcuni di essi, prevale il primo mezzo mentre nei rimanenti prevale il secondo. Quella che si ottiene, dunque, è una situazione abbastanza equilibrata dove il risultato del clustering si avvicina a quella reale, con entrambi i mezzi individuati.

La terza ed ultima settimana di registrazioni di dati GPS tratta dal dataset Geolife ed analizzata è il dataset GL3. I mezzi di trasporto contenuti in tale set sono quattro.

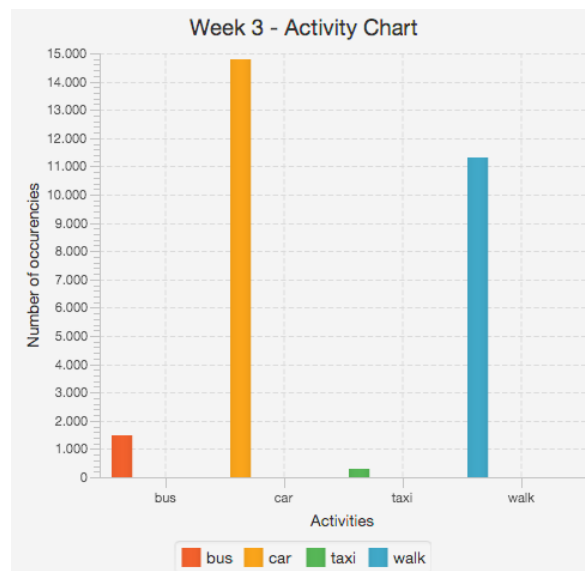


Figura 5.7: Grafico con la distribuzione dei mezzi di trasporto per il dataset GL3.

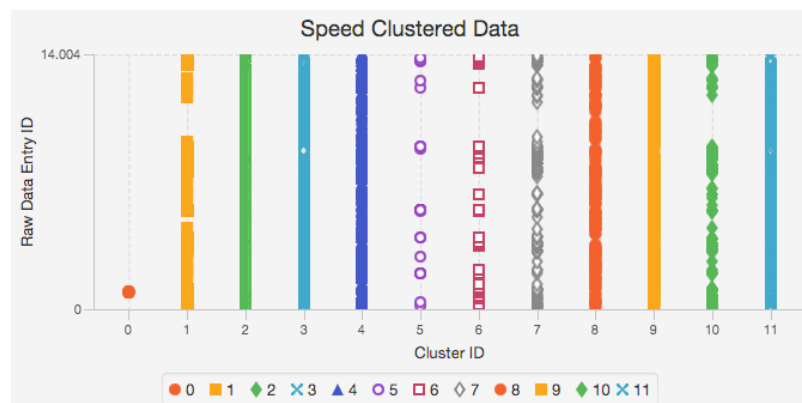


Figura 5.8: Grafico con l'esito del DPMM per il dataset GL3.

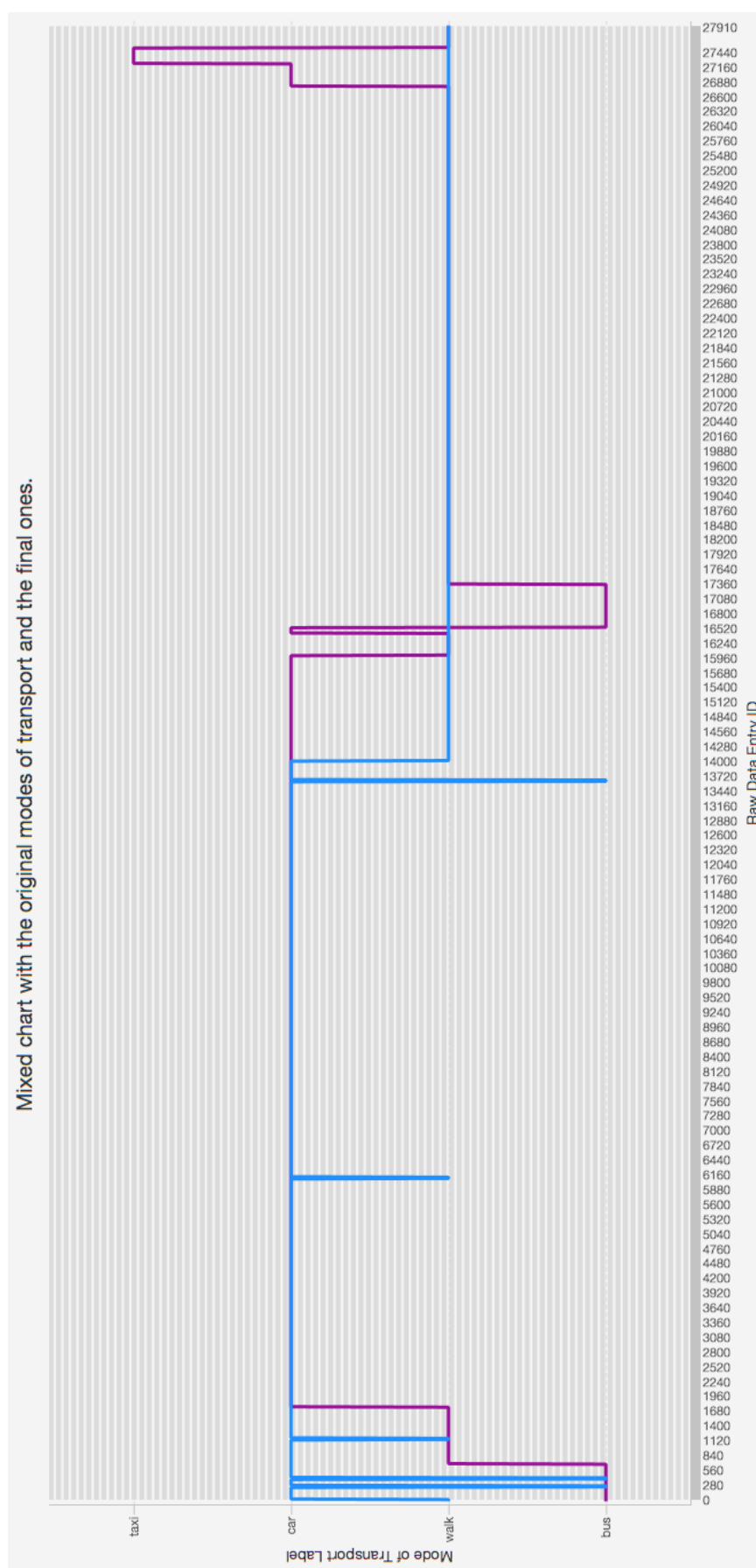


Figura 5.9: Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset GL3.

TOTAL NUMBER OF MODES OF TRANSPORT FOUND: 16

AMOUNT OF SUCCESS: 80.69415%

AMOUNT OF ERROR: 19.30585%

INSTANCES CORRECTLY CLUSTERED: 22529

INSTANCES INCORRECTLY CLUSTERED: 5390

MODE OF TRANSPORT STRUCTURE

MODE OF TRANSPORT ID: 0

LABEL bus – VALUE: 814 – PERCENTUAL: 5.836799%

LABEL car – VALUE: 2575 – PERCENTUAL: 18.464075%

LABEL taxi – VALUE: 306 – PERCENTUAL: 2.1941776%

LABEL walk – VALUE: 10251 – PERCENTUAL: 73.50495%

MODE OF TRANSPORT ID: 1

LABEL bus – VALUE: 60 – PERCENTUAL: 18.181818%

LABEL car – VALUE: 240 – PERCENTUAL: 72.72727%

LABEL walk – VALUE: 30 – PERCENTUAL: 9.090909%

MODE OF TRANSPORT ID: 2

LABEL car – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 3

LABEL car – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 4

LABEL bus – VALUE: 466 – PERCENTUAL: 3.9730582%

LABEL car – VALUE: 10394 – PERCENTUAL: 88.61796%

LABEL walk – VALUE: 869 – PERCENTUAL: 7.408986%

MODE OF TRANSPORT ID: 5

LABEL car – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 6

LABEL car – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 7

LABEL bus – VALUE: 30 – PERCENTUAL: 50.0%

LABEL car – VALUE: 30 – PERCENTUAL: 50.0%

MODE OF TRANSPORT ID: 8
LABEL bus – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 9
LABEL car – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 10
LABEL car – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 11
LABEL walk – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 12
LABEL car – VALUE: 60 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 13
LABEL car – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 14
LABEL car – VALUE: 30 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 15
LABEL bus – VALUE: 90 – PERCENTUAL: 6.0240965%
LABEL car – VALUE: 1254 – PERCENTUAL: 83.935745%
LABEL walk – VALUE: 150 – PERCENTUAL: 10.040161%

Sorgente 5.3: Struttura dei cluster individuati per il dataset GL3

Anche in questa terza settimana, come nella precedente, vengono individuati i due mezzi che occorrono di più nel dataset, che effettivamente costituiscono l'ottanta per cento del dataset stesso. Tuttavia, nonostante il valore di precisione finale sia elevato, il risultato non può essere definito soddisfacente poiché, a differenza della seconda settimana, vi sono ben due mezzi utilizzati dall'utente che vengono persi, a conferma che l'aumento degli stessi porta ad un risultato peggiore, a meno che essi non siano uniformemente distribuiti.

5.2.2 Twente

La caratteristica più interessante del dataset proveniente dall'università di Twente, oltre al fatto di essere costituito dai dati di accelerometro, magnetometro e giroscopio, è che i dati stessi siano stati registrati contemporaneamente da dispositivi tenuti su diverse parti del corpo. Si è scelto di mostrare, nel seguito, i risultati ottenuti analizzando quelli provenienti dal dispositivo contenuto nella tasca dei pantaloni dell'utente, poiché si tratta della posizione più probabile dove gli utenti portano il loro terminale. Il primo esempio mostra i risultati dell'analisi su tali dati effettuata considerando solamente i valori dell'accelerometro e fa riferimento al dataset TW1. Il numero di mezzi di trasporto è pari a cinque.

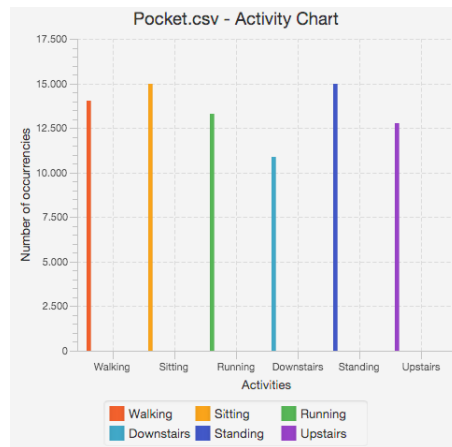


Figura 5.10: Grafico con la distribuzione dei mezzi di trasporto per il dataset TW1.

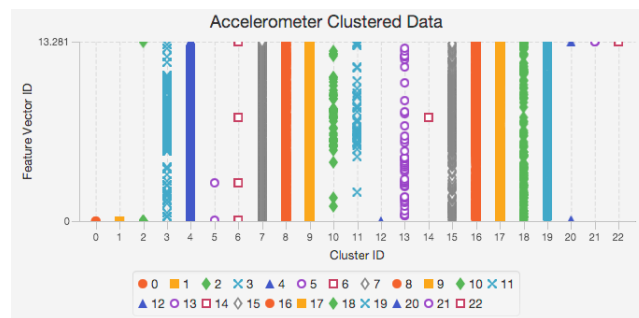


Figura 5.11: Grafico con l'esito del DPMM per il dataset TW1.

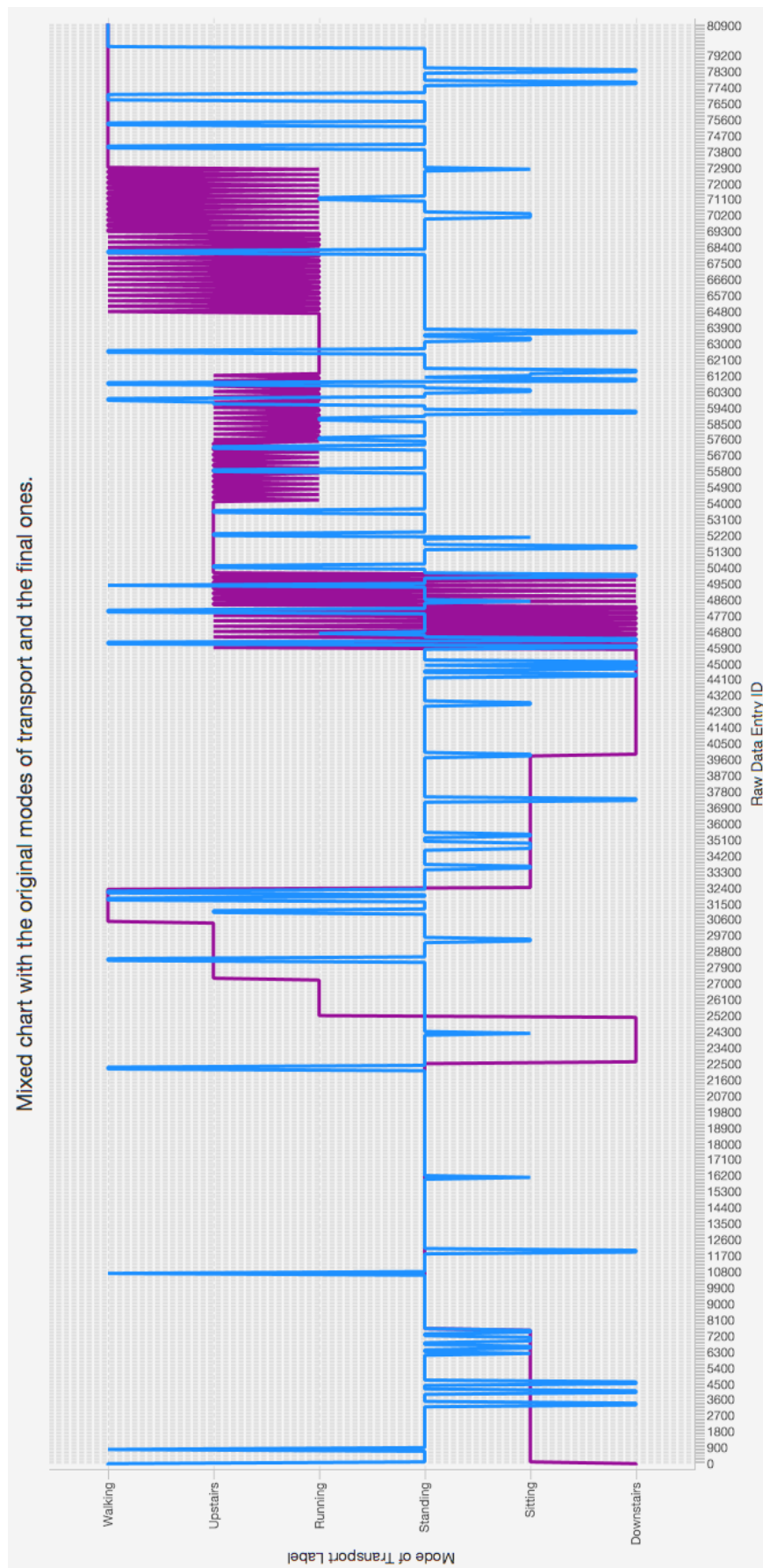


Figura 5.12: Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset TW1.

TOTAL NUMBER OF MODES OF TRANSPORT FOUND: 18

AMOUNT OF SUCCESS: 25.034887%

AMOUNT OF ERROR: 74.96511%

INSTANCES CORRECTLY CLUSTERED: 20273

INSTANCES INCORRECTLY CLUSTERED: 60706

MODE OF TRANSPORT STRUCTURE

MODE OF TRANSPORT ID: 0

LABEL Walking – VALUE: 1298 – PERCENTUAL: 87.762%

LABEL Downstairs – VALUE: 61 – PERCENTUAL: 4.1244082%

LABEL Upstairs – VALUE: 120 – PERCENTUAL: 8.11359%

MODE OF TRANSPORT ID: 1

LABEL Walking – VALUE: 360 – PERCENTUAL: 11.111112%

LABEL Sitting – VALUE: 720 – PERCENTUAL: 22.222223%

LABEL Running – VALUE: 600 – PERCENTUAL: 18.518518%

LABEL Downstairs – VALUE: 840 – PERCENTUAL: 25.925926%

LABEL Standing – VALUE: 180 – PERCENTUAL: 5.555556%

LABEL Upstairs – VALUE: 540 – PERCENTUAL: 16.666668%

MODE OF TRANSPORT ID: 2

LABEL Sitting – VALUE: 180 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 3

LABEL Walking – VALUE: 180 – PERCENTUAL: 33.333336%

LABEL Running – VALUE: 120 – PERCENTUAL: 22.222223%

LABEL Upstairs – VALUE: 240 – PERCENTUAL: 44.444447%

MODE OF TRANSPORT ID: 4

LABEL Walking – VALUE: 240 – PERCENTUAL: 26.666668%

LABEL Running – VALUE: 240 – PERCENTUAL: 26.666668%

LABEL Downstairs – VALUE: 120 – PERCENTUAL: 13.333334%

LABEL Standing – VALUE: 180 – PERCENTUAL: 20.0%

LABEL Upstairs – VALUE: 120 – PERCENTUAL: 13.333334%

MODE OF TRANSPORT ID: 5

LABEL Sitting – VALUE: 180 – PERCENTUAL: 50.0%

LABEL Downstairs – VALUE: 180 – PERCENTUAL: 50.0%

MODE OF TRANSPORT ID: 6

LABEL Walking – VALUE: 10553 – PERCENTUAL: 15.816847%
LABEL Sitting – VALUE: 12466 – PERCENTUAL: 18.684053%
LABEL Running – VALUE: 11012 – PERCENTUAL: 16.504795%
LABEL Downstairs – VALUE: 9112 – PERCENTUAL: 13.657074%
LABEL Standing – VALUE: 13501 – PERCENTUAL: 20.235312%
LABEL Upstairs – VALUE: 10076 – PERCENTUAL: 15.101918%

MODE OF TRANSPORT ID: 7

LABEL Running – VALUE: 120 – PERCENTUAL: 66.66667%
LABEL Upstairs – VALUE: 60 – PERCENTUAL: 33.333336%

MODE OF TRANSPORT ID: 8

LABEL Walking – VALUE: 120 – PERCENTUAL: 6.666667%
LABEL Sitting – VALUE: 540 – PERCENTUAL: 30.000002%
LABEL Running – VALUE: 324 – PERCENTUAL: 18.0%
LABEL Downstairs – VALUE: 180 – PERCENTUAL: 10.0%
LABEL Standing – VALUE: 180 – PERCENTUAL: 10.0%
LABEL Upstairs – VALUE: 456 – PERCENTUAL: 25.333332%

MODE OF TRANSPORT ID: 9

LABEL Upstairs – VALUE: 180 – PERCENTUAL: 100.0%
MOST FREQUENT LABEL: Upstairs – VALUE: 180

MODE OF TRANSPORT ID: 10

LABEL Walking – VALUE: 180 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 11

LABEL Walking – VALUE: 120 – PERCENTUAL: 22.222223%
LABEL Running – VALUE: 180 – PERCENTUAL: 33.333336%
LABEL Downstairs – VALUE: 120 – PERCENTUAL: 22.222223%
LABEL Upstairs – VALUE: 120 – PERCENTUAL: 22.222223%

MODE OF TRANSPORT ID: 12

LABEL Walking – VALUE: 254 – PERCENTUAL: 17.63889%
LABEL Sitting – VALUE: 613 – PERCENTUAL: 42.569443%
LABEL Running – VALUE: 286 – PERCENTUAL: 19.86111%
LABEL Downstairs – VALUE: 167 – PERCENTUAL: 11.597222%
LABEL Upstairs – VALUE: 120 – PERCENTUAL: 8.333334%

MODE OF TRANSPORT ID: 13

LABEL Standing – VALUE: 180 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 14

LABEL Walking – VALUE: 360 – PERCENTUAL: 25.0%

LABEL Sitting – VALUE: 180 – PERCENTUAL: 12.5%

LABEL Running – VALUE: 300 – PERCENTUAL: 20.833332%

LABEL Downstairs – VALUE: 120 – PERCENTUAL: 8.333334%

LABEL Standing – VALUE: 180 – PERCENTUAL: 12.5%

LABEL Upstairs – VALUE: 300 – PERCENTUAL: 20.833332%

MODE OF TRANSPORT ID: 15

LABEL Running – VALUE: 60 – PERCENTUAL: 33.333336%

LABEL Upstairs – VALUE: 120 – PERCENTUAL: 66.66667%

MODE OF TRANSPORT ID: 16

LABEL Standing – VALUE: 180 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 17

LABEL Sitting – VALUE: 121 – PERCENTUAL: 67.22222%

LABEL Standing – VALUE: 59 – PERCENTUAL: 32.77778%

MODE OF TRANSPORT ID: 18

LABEL Walking – VALUE: 360 – PERCENTUAL: 100.0%

Sorgente 5.4: Struttura dei cluster individuati per il dataset TW1

Il commento che si può fare a questo esempio è che l'utilizzo dei dati provenienti da un singolo sensore diverso dal GPS per effettuare il clustering è una pessima scelta; il livello di precisione finale è assolutamente insufficiente e osservando la struttura dei cluster individuati si può vedere come ci sia molta confusione nella composizione degli stessi da parte degli algoritmi utilizzati.

Per quanto riguarda il secondo esempio illustrato nel seguito, che fa riferimento al dataset TW4, la tipologia dei dati utilizzati ed il numero dei mezzi di trasporto sono gli stessi dell'esempio precedente. La differenza sta nel fatto che vengono sfruttati i dati di due sensori, accelerometro e giroscopio, nel processo di clustering.

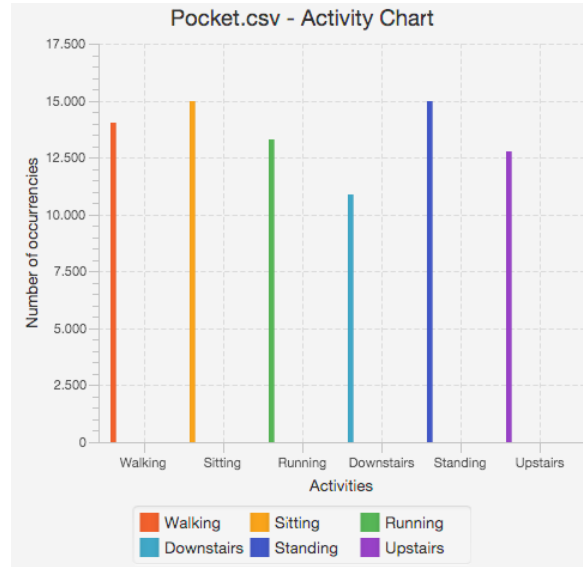


Figura 5.13: Grafico con la distribuzione dei mezzi di trasporto per il dataset TW4.

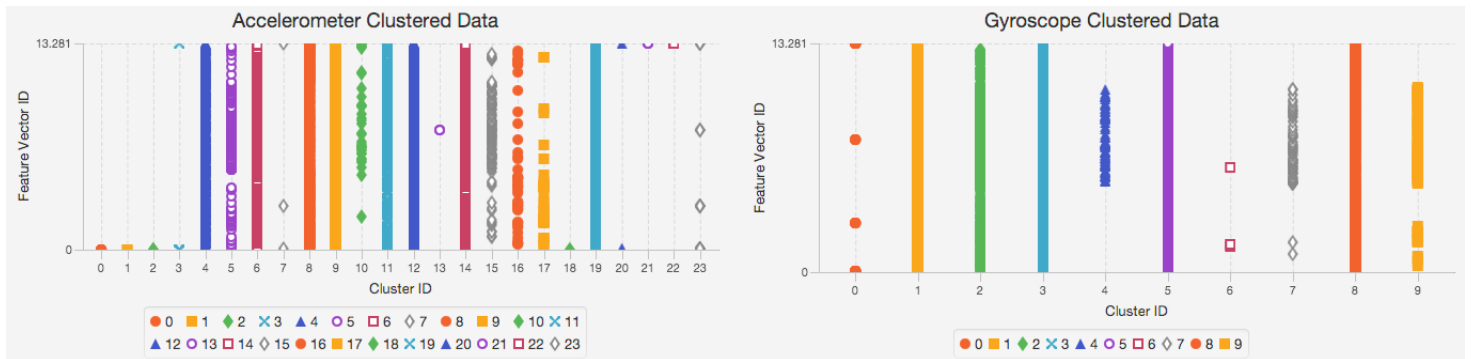


Figura 5.14: Grafico con l'esito del DPMM per il dataset TW4.

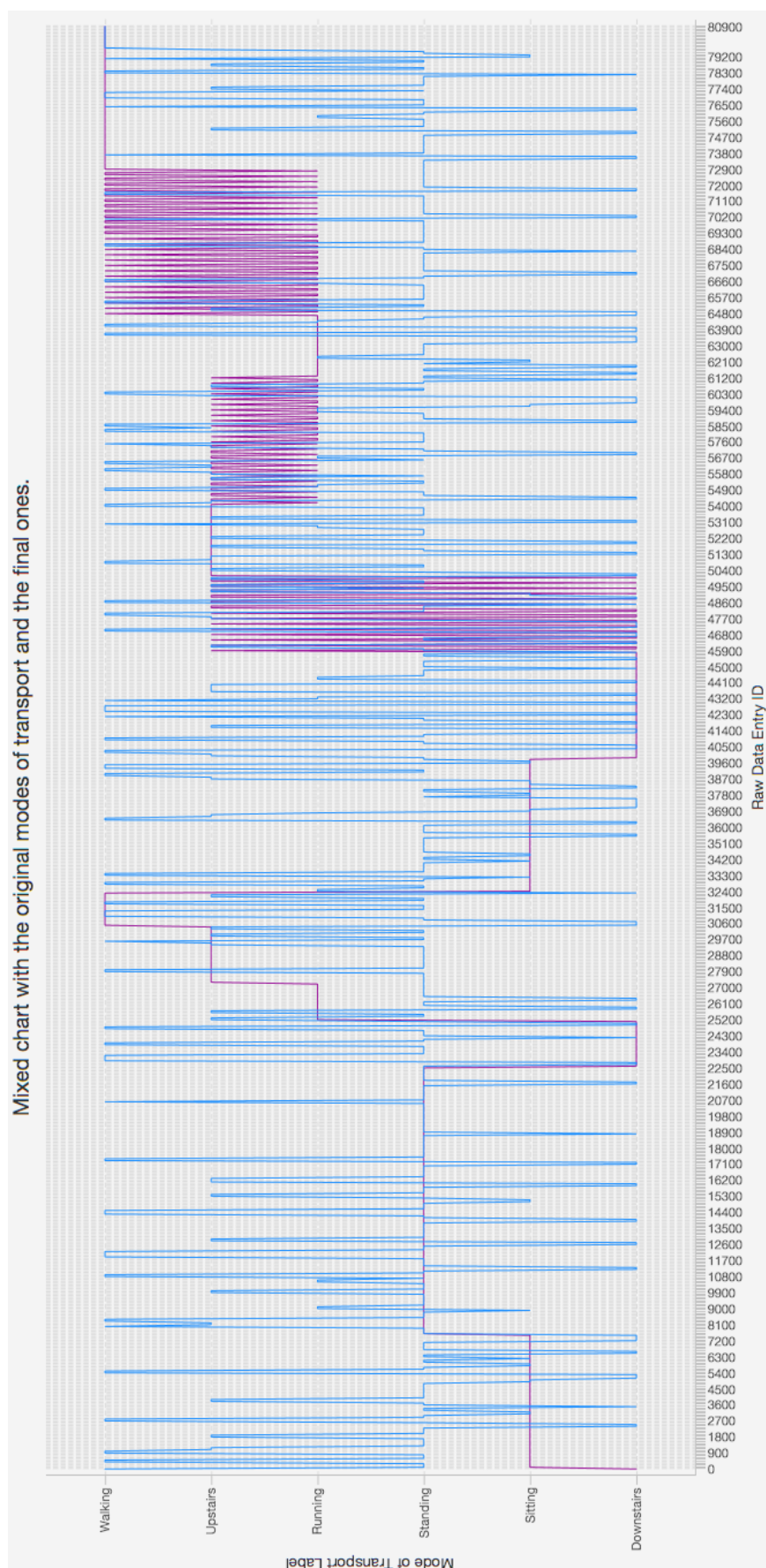


Figura 5.15: Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset TW4.

TOTAL NUMBER OF MODES OF TRANSPORT FOUND: 21

AMOUNT OF SUCCESS: 27.866482%

AMOUNT OF ERROR: 72.133514%

INSTANCES CORRECTLY CLUSTERED: 22566

INSTANCES INCORRECTLY CLUSTERED: 58413

MODE OF TRANSPORT STRUCTURE

MODE OF TRANSPORT ID: 0

LABEL Walking – VALUE: 1358 – PERCENTUAL: 57.08281%

LABEL Sitting – VALUE: 360 – PERCENTUAL: 15.132408%

LABEL Running – VALUE: 370 – PERCENTUAL: 15.552753%

LABEL Downstairs – VALUE: 5 – PERCENTUAL: 0.21017234%

LABEL Upstairs – VALUE: 286 – PERCENTUAL: 12.021858%

MODE OF TRANSPORT ID: 1

LABEL Downstairs – VALUE: 120 – PERCENTUAL: 66.66667%

LABEL Upstairs – VALUE: 60 – PERCENTUAL: 33.333336%

MODE OF TRANSPORT ID: 2

LABEL Running – VALUE: 60 – PERCENTUAL: 33.333336%

LABEL Upstairs – VALUE: 120 – PERCENTUAL: 66.66667%

MODE OF TRANSPORT ID: 3

LABEL Running – VALUE: 120 – PERCENTUAL: 66.66667%

LABEL Upstairs – VALUE: 60 – PERCENTUAL: 33.333336%

MODE OF TRANSPORT ID: 4

LABEL Sitting – VALUE: 180 – PERCENTUAL: 25.0%

LABEL Running – VALUE: 120 – PERCENTUAL: 16.666668%

LABEL Downstairs – VALUE: 179 – PERCENTUAL: 24.86111%

LABEL Standing – VALUE: 1 – PERCENTUAL: 0.1388889%

LABEL Upstairs – VALUE: 240 – PERCENTUAL: 33.333336%

MODE OF TRANSPORT ID: 5

LABEL Walking – VALUE: 7266 – PERCENTUAL: 16.409584%

LABEL Sitting – VALUE: 8447 – PERCENTUAL: 19.076763%

LABEL Running – VALUE: 7143 – PERCENTUAL: 16.1318%

LABEL Downstairs – VALUE: 4909 – PERCENTUAL: 11.08652%

LABEL Standing – VALUE: 9959 – PERCENTUAL: 22.491474%
LABEL Upstairs – VALUE: 6555 – PERCENTUAL: 14.803858%

MODE OF TRANSPORT ID: 6

LABEL Sitting – VALUE: 180 – PERCENTUAL: 12.5%
LABEL Running – VALUE: 300 – PERCENTUAL: 20.833332%
LABEL Downstairs – VALUE: 480 – PERCENTUAL: 33.333336%
LABEL Upstairs – VALUE: 480 – PERCENTUAL: 33.333336%

MODE OF TRANSPORT ID: 7

LABEL Downstairs – VALUE: 360 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 8

LABEL Walking – VALUE: 180 – PERCENTUAL: 11.111112%
LABEL Sitting – VALUE: 540 – PERCENTUAL: 33.333336%
LABEL Running – VALUE: 180 – PERCENTUAL: 11.111112%
LABEL Downstairs – VALUE: 360 – PERCENTUAL: 22.222223%
LABEL Standing – VALUE: 180 – PERCENTUAL: 11.111112%
LABEL Upstairs – VALUE: 180 – PERCENTUAL: 11.111112%

MODE OF TRANSPORT ID: 9

LABEL Walking – VALUE: 480 – PERCENTUAL: 29.62963%
LABEL Running – VALUE: 240 – PERCENTUAL: 14.814815%
LABEL Downstairs – VALUE: 180 – PERCENTUAL: 11.111112%
LABEL Standing – VALUE: 720 – PERCENTUAL: 44.444447%

MODE OF TRANSPORT ID: 10

LABEL Walking – VALUE: 1381 – PERCENTUAL: 27.725357%
LABEL Sitting – VALUE: 1800 – PERCENTUAL: 36.13732%
LABEL Running – VALUE: 660 – PERCENTUAL: 13.250351%
LABEL Downstairs – VALUE: 360 – PERCENTUAL: 7.227464%
LABEL Standing – VALUE: 360 – PERCENTUAL: 7.227464%
LABEL Upstairs – VALUE: 420 – PERCENTUAL: 8.432042%

MODE OF TRANSPORT ID: 11

LABEL Walking – VALUE: 540 – PERCENTUAL: 33.333336%
LABEL Sitting – VALUE: 180 – PERCENTUAL: 11.111112%
LABEL Running – VALUE: 420 – PERCENTUAL: 25.925926%
LABEL Downstairs – VALUE: 180 – PERCENTUAL: 11.111112%
LABEL Standing – VALUE: 180 – PERCENTUAL: 11.111112%

LABEL Upstairs – VALUE: 120 – PERCENTUAL: 7.4074073%

MODE OF TRANSPORT ID: 12

LABEL Walking – VALUE: 180 – PERCENTUAL: 25.0%

LABEL Sitting – VALUE: 180 – PERCENTUAL: 25.0%

LABEL Downstairs – VALUE: 120 – PERCENTUAL: 16.666668%

LABEL Upstairs – VALUE: 240 – PERCENTUAL: 33.333336%

MODE OF TRANSPORT ID: 13

LABEL Walking – VALUE: 120 – PERCENTUAL: 66.66667%

LABEL Running – VALUE: 60 – PERCENTUAL: 33.333336%

MODE OF TRANSPORT ID: 14

LABEL Walking – VALUE: 720 – PERCENTUAL: 15.384616%

LABEL Sitting – VALUE: 1080 – PERCENTUAL: 23.076923%

LABEL Running – VALUE: 900 – PERCENTUAL: 19.23077%

LABEL Downstairs – VALUE: 1080 – PERCENTUAL: 23.076923%

LABEL Standing – VALUE: 540 – PERCENTUAL: 11.538462%

LABEL Upstairs – VALUE: 360 – PERCENTUAL: 7.692308%

MODE OF TRANSPORT ID: 15

LABEL Walking – VALUE: 31 – PERCENTUAL: 17.222223%

LABEL Running – VALUE: 149 – PERCENTUAL: 82.77778%

MODE OF TRANSPORT ID: 16

LABEL Walking – VALUE: 180 – PERCENTUAL: 7.692308%

LABEL Sitting – VALUE: 360 – PERCENTUAL: 15.384616%

LABEL Running – VALUE: 480 – PERCENTUAL: 20.512821%

LABEL Downstairs – VALUE: 660 – PERCENTUAL: 28.20513%

LABEL Standing – VALUE: 360 – PERCENTUAL: 15.384616%

LABEL Upstairs – VALUE: 300 – PERCENTUAL: 12.820514%

MODE OF TRANSPORT ID: 17

LABEL Standing – VALUE: 180 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 18

LABEL Sitting – VALUE: 180 – PERCENTUAL: 11.111112%

LABEL Running – VALUE: 540 – PERCENTUAL: 33.333336%

LABEL Downstairs – VALUE: 120 – PERCENTUAL: 7.4074073%

LABEL Upstairs – VALUE: 780 – PERCENTUAL: 48.148148%

MODE OF TRANSPORT ID: 19

LABEL Running – VALUE: 180 – PERCENTUAL: 14.285715%

LABEL Downstairs – VALUE: 660 – PERCENTUAL: 52.380955%

LABEL Standing – VALUE: 180 – PERCENTUAL: 14.285715%

LABEL Upstairs – VALUE: 240 – PERCENTUAL: 19.047619%

MODE OF TRANSPORT ID: 20

LABEL Walking – VALUE: 1589 – PERCENTUAL: 15.4873295%

LABEL Sitting – VALUE: 1513 – PERCENTUAL: 14.746589%

LABEL Running – VALUE: 1380 – PERCENTUAL: 13.450292%

LABEL Downstairs – VALUE: 1127 – PERCENTUAL: 10.9844055%

LABEL Standing – VALUE: 2340 – PERCENTUAL: 22.807016%

LABEL Upstairs – VALUE: 2311 – PERCENTUAL: 22.524366%

Sorgente 5.5: Struttura dei cluster individuati per il dataset TW4

Per questo esempio rimangono valide le considerazioni fatte per quello precedente. Nonostante l'aggiunta di un ulteriore sensore abbia portato ad un leggero miglioramento delle prestazioni, il risultato finale è ancora molto insoddisfacente.

Anche per il terzo ed ultimo esempio, che fa riferimento al dataset TW5, la tipologia dei dati utilizzati ed il numero dei mezzi di trasporto sono analoghi a quelli del primo; il valore aggiunto sta nel fatto che vengono utilizzati i dati di tutte e tre i sensori disponibili per il dataset.

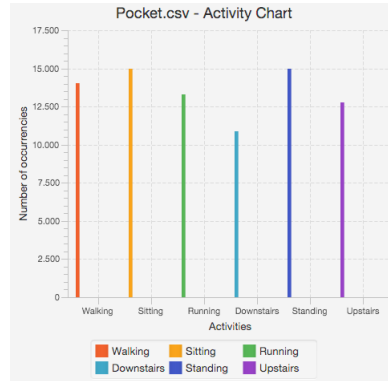


Figura 5.16: Grafico con la distribuzione dei mezzi di trasporto per il dataset TW5.

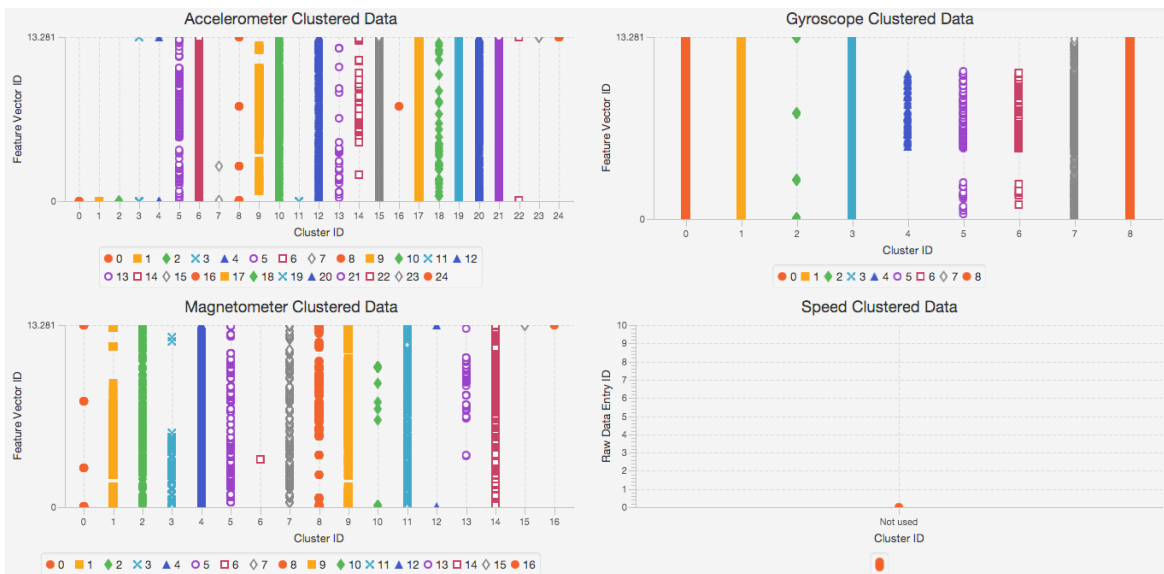


Figura 5.17: Grafico con l'esito del DPMM per il dataset TW5.

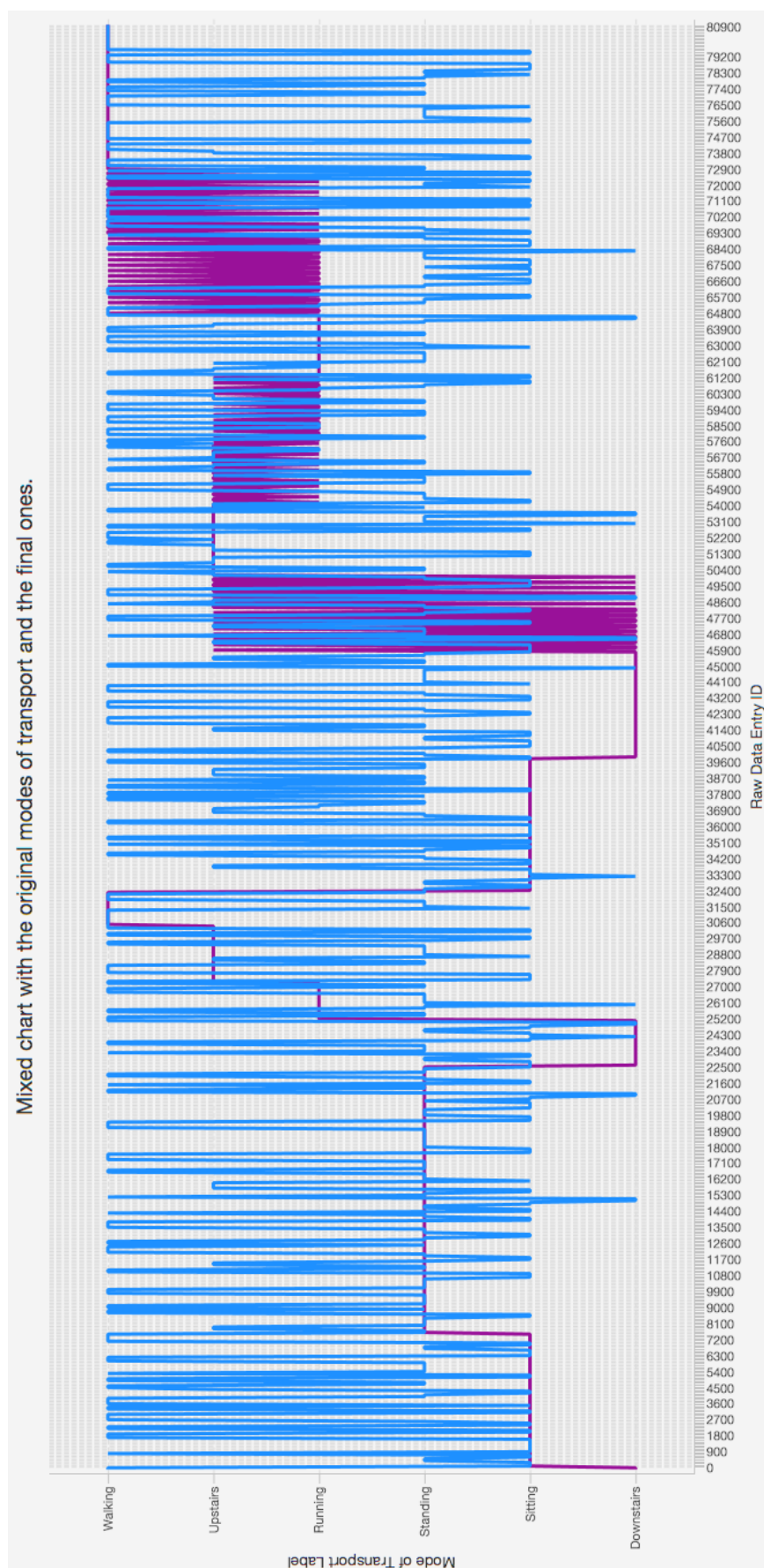


Figura 5.18: Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset TW5.

TOTAL NUMBER OF MODES OF TRANSPORT FOUND: 26

AMOUNT OF SUCCESS: 29.90652%

AMOUNT OF ERROR: 70.093475%

INSTANCES CORRECTLY CLUSTERED: 24218

INSTANCES INCORRECTLY CLUSTERED: 56761

MODE OF TRANSPORT STRUCTURE

MODE OF TRANSPORT ID: 0

LABEL Walking – VALUE: 1759 – PERCENTUAL: 50.852848%

LABEL Sitting – VALUE: 360 – PERCENTUAL: 10.407633%

LABEL Running – VALUE: 379 – PERCENTUAL: 10.9569235%

LABEL Downstairs – VALUE: 61 – PERCENTUAL: 1.7635156%

LABEL Standing – VALUE: 360 – PERCENTUAL: 10.407633%

LABEL Upstairs – VALUE: 540 – PERCENTUAL: 15.611449%

MODE OF TRANSPORT ID: 1

LABEL Walking – VALUE: 180 – PERCENTUAL: 6.666667%

LABEL Sitting – VALUE: 180 – PERCENTUAL: 6.666667%

LABEL Running – VALUE: 240 – PERCENTUAL: 8.888889%

LABEL Downstairs – VALUE: 840 – PERCENTUAL: 31.111113%

LABEL Standing – VALUE: 900 – PERCENTUAL: 33.333336%

LABEL Upstairs – VALUE: 360 – PERCENTUAL: 13.333334%

MODE OF TRANSPORT ID: 2

LABEL Sitting – VALUE: 539 – PERCENTUAL: 49.95366%

LABEL Downstairs – VALUE: 180 – PERCENTUAL: 16.682114%

LABEL Standing – VALUE: 360 – PERCENTUAL: 33.364227%

MODE OF TRANSPORT ID: 3

LABEL Walking – VALUE: 360 – PERCENTUAL: 10.0%

LABEL Sitting – VALUE: 360 – PERCENTUAL: 10.0%

LABEL Running – VALUE: 660 – PERCENTUAL: 18.333334%

LABEL Downstairs – VALUE: 360 – PERCENTUAL: 10.0%

LABEL Standing – VALUE: 1440 – PERCENTUAL: 40.0%

LABEL Upstairs – VALUE: 420 – PERCENTUAL: 11.666667%

MODE OF TRANSPORT ID: 4

LABEL Walking – VALUE: 1091 – PERCENTUAL: 12.369615%

LABEL Sitting – VALUE: 1667 – PERCENTUAL: 18.900227%
LABEL Running – VALUE: 1680 – PERCENTUAL: 19.047619%
LABEL Downstairs – VALUE: 1326 – PERCENTUAL: 15.034014%
LABEL Standing – VALUE: 2160 – PERCENTUAL: 24.489796%
LABEL Upstairs – VALUE: 896 – PERCENTUAL: 10.1587305%

MODE OF TRANSPORT ID: 5

LABEL Walking – VALUE: 960 – PERCENTUAL: 15.238096%
LABEL Sitting – VALUE: 900 – PERCENTUAL: 14.285715%
LABEL Running – VALUE: 1080 – PERCENTUAL: 17.142859%
LABEL Downstairs – VALUE: 840 – PERCENTUAL: 13.333334%
LABEL Standing – VALUE: 1260 – PERCENTUAL: 20.0%
LABEL Upstairs – VALUE: 1260 – PERCENTUAL: 20.0%

MODE OF TRANSPORT ID: 6

LABEL Downstairs – VALUE: 60 – PERCENTUAL: 33.333336%
LABEL Upstairs – VALUE: 120 – PERCENTUAL: 66.66667%

MODE OF TRANSPORT ID: 7

LABEL Walking – VALUE: 60 – PERCENTUAL: 4.7619047%
LABEL Sitting – VALUE: 180 – PERCENTUAL: 14.285715%
LABEL Running – VALUE: 300 – PERCENTUAL: 23.809525%
LABEL Downstairs – VALUE: 360 – PERCENTUAL: 28.57143%
LABEL Upstairs – VALUE: 360 – PERCENTUAL: 28.57143%

MODE OF TRANSPORT ID: 8

LABEL Running – VALUE: 300 – PERCENTUAL: 55.555557%
LABEL Upstairs – VALUE: 240 – PERCENTUAL: 44.444447%

MODE OF TRANSPORT ID: 9

LABEL Walking – VALUE: 180 – PERCENTUAL: 12.5%
LABEL Sitting – VALUE: 180 – PERCENTUAL: 12.5%
LABEL Running – VALUE: 360 – PERCENTUAL: 25.0%
LABEL Standing – VALUE: 720 – PERCENTUAL: 50.0%

MODE OF TRANSPORT ID: 10

LABEL Walking – VALUE: 3314 – PERCENTUAL: 24.548147%
LABEL Sitting – VALUE: 2641 – PERCENTUAL: 19.562963%
LABEL Running – VALUE: 2817 – PERCENTUAL: 20.866667%
LABEL Downstairs – VALUE: 1439 – PERCENTUAL: 10.65926%

LABEL Standing – VALUE: 1859 – PERCENTUAL: 13.7703705%
LABEL Upstairs – VALUE: 1430 – PERCENTUAL: 10.592592%

MODE OF TRANSPORT ID: 11

LABEL Sitting – VALUE: 180 – PERCENTUAL: 20.0%
LABEL Running – VALUE: 240 – PERCENTUAL: 26.666668%
LABEL Downstairs – VALUE: 120 – PERCENTUAL: 13.333334%
LABEL Upstairs – VALUE: 360 – PERCENTUAL: 40.0%

MODE OF TRANSPORT ID: 12

LABEL Running – VALUE: 221 – PERCENTUAL: 30.694443%
LABEL Downstairs – VALUE: 4 – PERCENTUAL: 0.555556%
LABEL Upstairs – VALUE: 495 – PERCENTUAL: 68.75%

MODE OF TRANSPORT ID: 13

LABEL Walking – VALUE: 1260 – PERCENTUAL: 22.580645%
LABEL Sitting – VALUE: 900 – PERCENTUAL: 16.129032%
LABEL Running – VALUE: 900 – PERCENTUAL: 16.129032%
LABEL Downstairs – VALUE: 600 – PERCENTUAL: 10.752688%
LABEL Standing – VALUE: 900 – PERCENTUAL: 16.129032%
LABEL Upstairs – VALUE: 1020 – PERCENTUAL: 18.27957%

MODE OF TRANSPORT ID: 14

LABEL Walking – VALUE: 180 – PERCENTUAL: 10.0%
LABEL Sitting – VALUE: 360 – PERCENTUAL: 20.0%
LABEL Running – VALUE: 352 – PERCENTUAL: 19.555555%
LABEL Standing – VALUE: 360 – PERCENTUAL: 20.0%
LABEL Upstairs – VALUE: 548 – PERCENTUAL: 30.444443%

MODE OF TRANSPORT ID: 15

LABEL Walking – VALUE: 31 – PERCENTUAL: 4.305556%
LABEL Running – VALUE: 149 – PERCENTUAL: 20.694445%
LABEL Downstairs – VALUE: 180 – PERCENTUAL: 25.0%
LABEL Standing – VALUE: 180 – PERCENTUAL: 25.0%
LABEL Upstairs – VALUE: 180 – PERCENTUAL: 25.0%

MODE OF TRANSPORT ID: 16

LABEL Walking – VALUE: 180 – PERCENTUAL: 14.285715%
LABEL Sitting – VALUE: 540 – PERCENTUAL: 42.857143%
LABEL Downstairs – VALUE: 360 – PERCENTUAL: 28.57143%

LABEL Standing – VALUE: 180 – PERCENTUAL: 14.285715%

MODE OF TRANSPORT ID: 17

LABEL Sitting – VALUE: 540 – PERCENTUAL: 21.428572%

LABEL Running – VALUE: 420 – PERCENTUAL: 16.666668%

LABEL Downstairs – VALUE: 480 – PERCENTUAL: 19.047619%

LABEL Standing – VALUE: 360 – PERCENTUAL: 14.285715%

LABEL Upstairs – VALUE: 720 – PERCENTUAL: 28.57143%

MODE OF TRANSPORT ID: 18

LABEL Walking – VALUE: 60 – PERCENTUAL: 33.333336%

LABEL Running – VALUE: 120 – PERCENTUAL: 66.66667%

MODE OF TRANSPORT ID: 19

LABEL Sitting – VALUE: 360 – PERCENTUAL: 28.57143%

LABEL Running – VALUE: 120 – PERCENTUAL: 9.523809%

LABEL Downstairs – VALUE: 360 – PERCENTUAL: 28.57143%

LABEL Standing – VALUE: 180 – PERCENTUAL: 14.285715%

LABEL Upstairs – VALUE: 240 – PERCENTUAL: 19.047619%

MODE OF TRANSPORT ID: 20

LABEL Walking – VALUE: 60 – PERCENTUAL: 6.666667%

LABEL Sitting – VALUE: 180 – PERCENTUAL: 20.0%

LABEL Running – VALUE: 120 – PERCENTUAL: 13.333334%

LABEL Downstairs – VALUE: 180 – PERCENTUAL: 20.0%

LABEL Standing – VALUE: 360 – PERCENTUAL: 40.0%

MODE OF TRANSPORT ID: 21

LABEL Walking – VALUE: 2640 – PERCENTUAL: 20.37037%

LABEL Sitting – VALUE: 2880 – PERCENTUAL: 22.222223%

LABEL Running – VALUE: 1764 – PERCENTUAL: 13.611111%

LABEL Downstairs – VALUE: 1904 – PERCENTUAL: 14.691358%

LABEL Standing – VALUE: 1980 – PERCENTUAL: 15.277778%

LABEL Upstairs – VALUE: 1792 – PERCENTUAL: 13.82716%

MODE OF TRANSPORT ID: 22

LABEL Walking – VALUE: 241 – PERCENTUAL: 36.459908%

LABEL Sitting – VALUE: 180 – PERCENTUAL: 27.231466%

LABEL Running – VALUE: 60 – PERCENTUAL: 9.077156%

LABEL Standing – VALUE: 180 – PERCENTUAL: 27.231466%

MODE OF TRANSPORT ID: 23

LABEL Walking – VALUE: 1289 – PERCENTUAL: 31.135267%

LABEL Sitting – VALUE: 540 – PERCENTUAL: 13.043478%

LABEL Running – VALUE: 420 – PERCENTUAL: 10.144927%

LABEL Downstairs – VALUE: 420 – PERCENTUAL: 10.144927%

LABEL Standing – VALUE: 360 – PERCENTUAL: 8.695652%

LABEL Upstairs – VALUE: 1111 – PERCENTUAL: 26.835749%

MODE OF TRANSPORT ID: 24

LABEL Running – VALUE: 60 – PERCENTUAL: 33.333336%

LABEL Upstairs – VALUE: 120 – PERCENTUAL: 66.666667%

MODE OF TRANSPORT ID: 25

LABEL Downstairs – VALUE: 60 – PERCENTUAL: 16.666668%

LABEL Standing – VALUE: 180 – PERCENTUAL: 50.0%

LABEL Upstairs – VALUE: 120 – PERCENTUAL: 33.333336%

MODE OF TRANSPORT ID: 26

LABEL Sitting – VALUE: 180 – PERCENTUAL: 100.0%

Sorgente 5.6: Struttura dei cluster individuati per il dataset TW5

Anche in questo caso non cambia nulla rispetto a quanto detto per i due esempi precedenti, se non che si verifica un ulteriore aumento delle prestazioni. Da tale considerazione dai test effettuati su Geolife, si deduce, dunque, che l'utilizzo combinato di tutti e quattro i sensori può portare ad un miglioramento generale non solo nella precisione del risultato ma anche nella struttura dei cluster individuati dal processo.

5.2.3 Lifetracker

Per quanto riguarda il dataset in-house, si è deciso di mostrare nel seguito i risultati ottenuti su una sola giornata di registrazione e successivamente per un'intera settimana applicando il processo di clustering ai dati di tutti i sensori disponibili. Il primo esempio fa riferimento al dataset IH8 ed è caratterizzato da tre mezzi di trasporto.

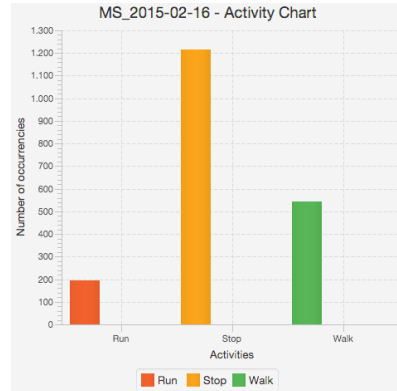


Figura 5.19: Grafico con la distribuzione dei mezzi di trasporto per dataset IH8.

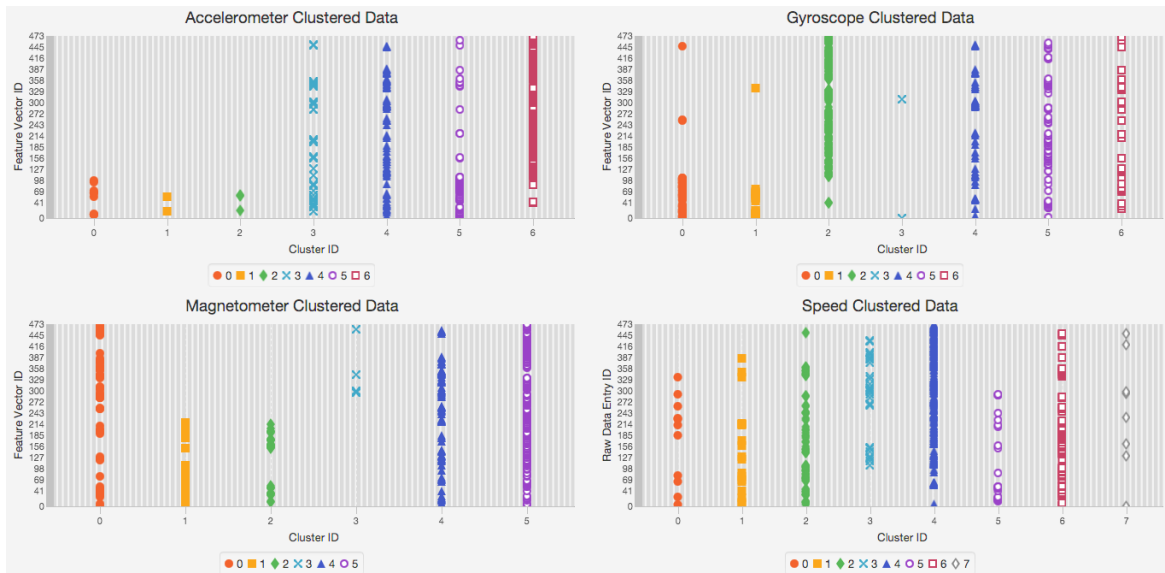


Figura 5.20: Grafico con l'esito del DPMM per il dataset IH8.

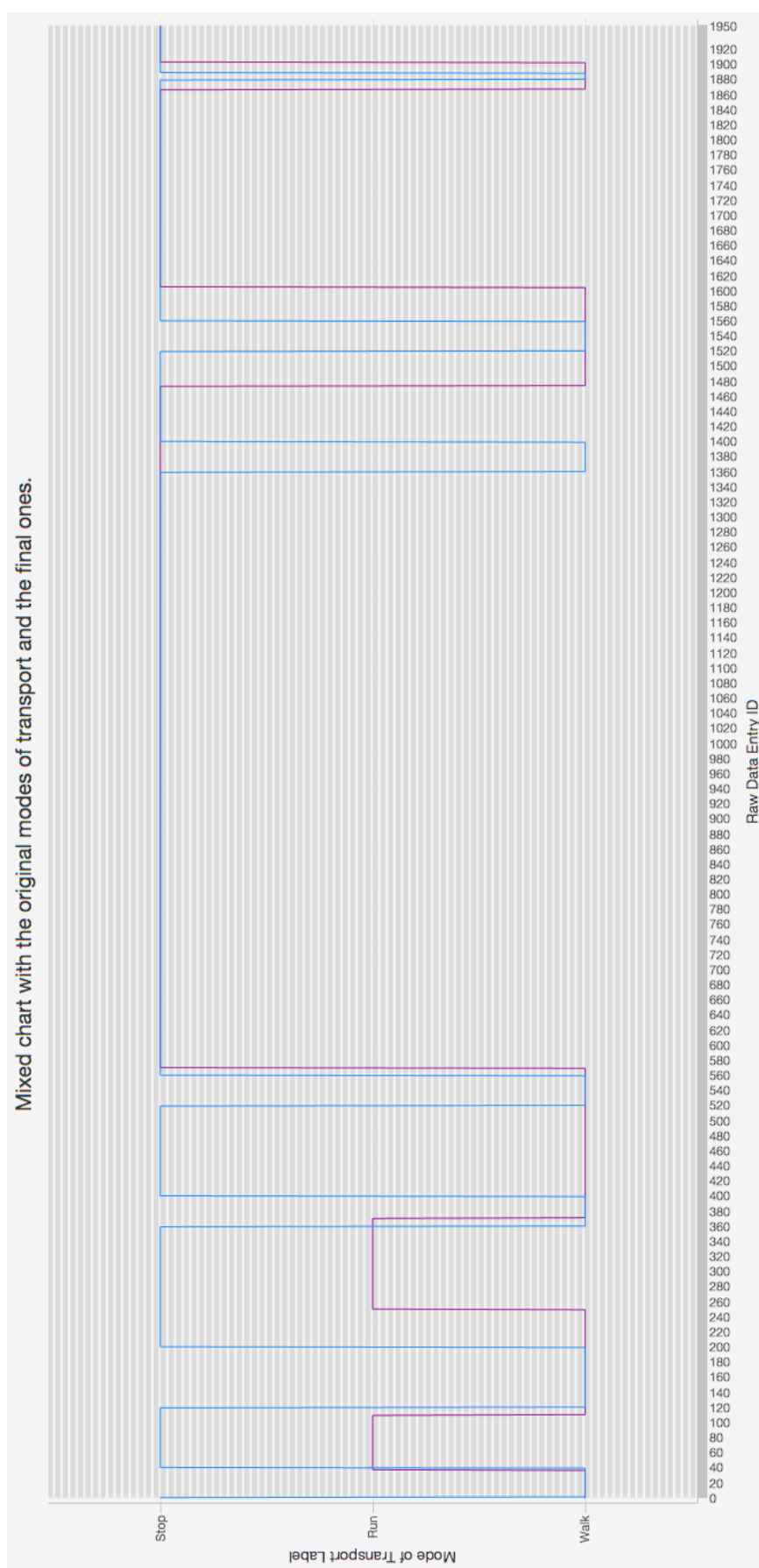


Figura 5.21: Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset IH8.

TOTAL NUMBER OF MODES OF TRANSPORT FOUND: 6

AMOUNT OF SUCCESS: 72.16812%

AMOUNT OF ERROR: 27.831882%

INSTANCES CORRECTLY CLUSTERED: 1408

INSTANCES INCORRECTLY CLUSTERED: 543

MODE OF TRANSPORT STRUCTURE

MODE OF TRANSPORT ID: 0

LABEL r – VALUE: 40 – PERCENTUAL: 11.661807%

LABEL s – VALUE: 243 – PERCENTUAL: 70.84548%

LABEL w – VALUE: 60 – PERCENTUAL: 17.492712%

MODE OF TRANSPORT ID: 1

LABEL w – VALUE: 40 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 2

LABEL r – VALUE: 70 – PERCENTUAL: 29.166666%

LABEL s – VALUE: 114 – PERCENTUAL: 47.5%

LABEL w – VALUE: 56 – PERCENTUAL: 23.333334%

MODE OF TRANSPORT ID: 3

LABEL r – VALUE: 11 – PERCENTUAL: 6.875%

LABEL s – VALUE: 40 – PERCENTUAL: 25.0%

LABEL w – VALUE: 109 – PERCENTUAL: 68.125%

MODE OF TRANSPORT ID: 4

LABEL r – VALUE: 3 – PERCENTUAL: 3.409091%

LABEL w – VALUE: 85 – PERCENTUAL: 96.590904%

MODE OF TRANSPORT ID: 5

LABEL r – VALUE: 70 – PERCENTUAL: 6.4814816%

LABEL s – VALUE: 817 – PERCENTUAL: 75.64815%

LABEL w – VALUE: 193 – PERCENTUAL: 17.87037%

Sorgente 5.7: Struttura dei cluster individuati per il dataset IH8

Per questo esempio, il risultato finale si può considerare soddisfacente. La precisione è piuttosto elevata e vengono riconosciuti due mezzi su tre, a conferma del fatto che utilizzare tutti e quattro i sensori porta ad un miglioramento generale delle prestazioni del processo di clustering.

Il secondo ed ultimo esempio mostra i risultati del processo di clustering applicato al dataset IH1 con il feature set otto. Si tratta di una settimana di registrazione, dove sono stati utilizzati cinque mezzi di trasporto.

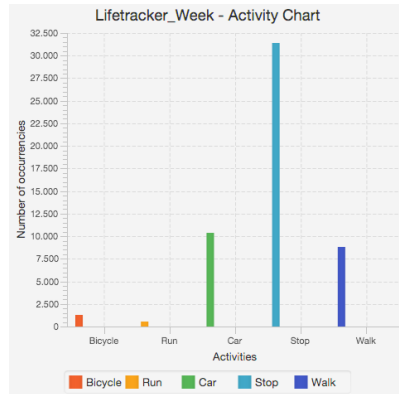


Figura 5.22: Grafico con la distribuzione dei mezzi di trasporto per il dataset IH1 con il feature set otto.

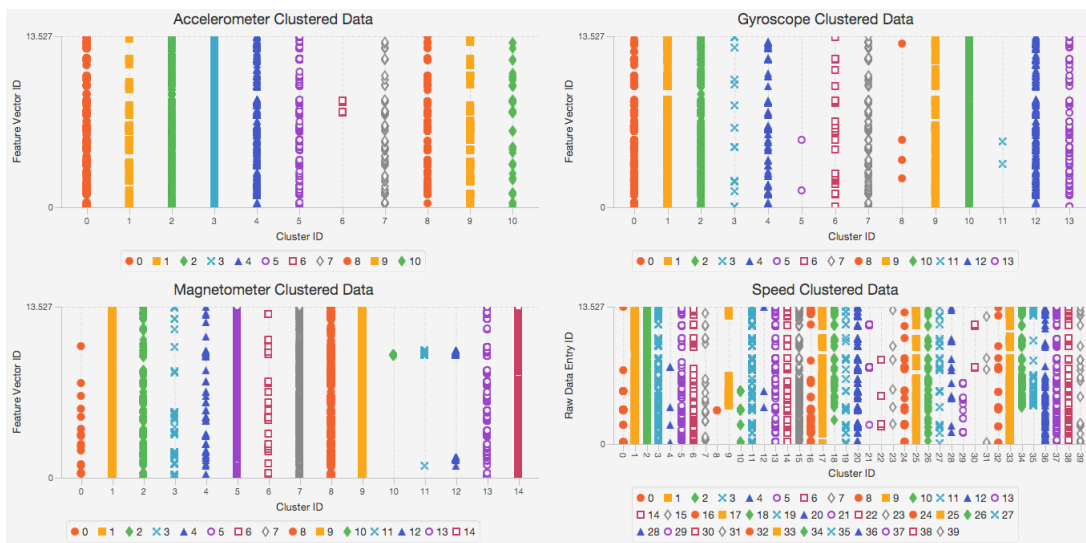


Figura 5.23: Grafico con l'esito del DPMM per il dataset IH1 con il feature set otto.

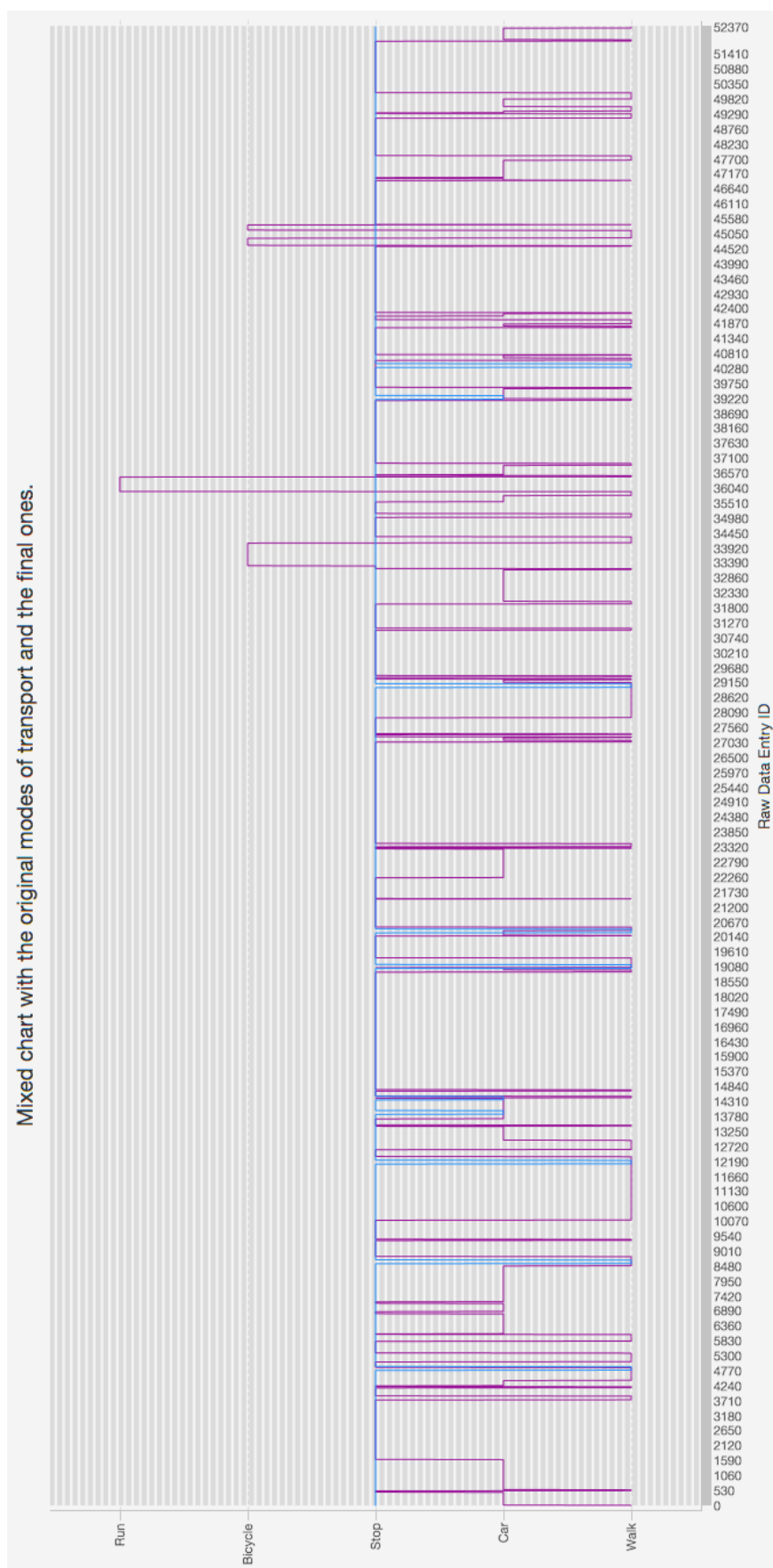


Figura 5.24: Grafico con il confronto tra gli assegnamenti originali e quelli finali per il dataset IH1 con il feature set otto.

TOTAL NUMBER OF MODES OF TRANSPORT FOUND: 21

AMOUNT OF SUCCESS: 61.320198%

AMOUNT OF ERROR: 38.679802%

INSTANCES CORRECTLY CLUSTERED: 32114

INSTANCES INCORRECTLY CLUSTERED: 20257

MODE OF TRANSPORT STRUCTURE

MODE OF TRANSPORT ID: 0

LABEL bc – VALUE: 443 – PERCENTUAL: 3.7145734%

LABEL c – VALUE: 1825 – PERCENTUAL: 15.3027%

LABEL s – VALUE: 8229 – PERCENTUAL: 69.0005%

LABEL w – VALUE: 1429 – PERCENTUAL: 11.9822235%

MODE OF TRANSPORT ID: 1

LABEL w – VALUE: 126 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 2

LABEL s – VALUE: 31 – PERCENTUAL: 24.603174%

LABEL w – VALUE: 95 – PERCENTUAL: 75.39682%

MODE OF TRANSPORT ID: 3

LABEL c – VALUE: 81 – PERCENTUAL: 16.071428%

LABEL s – VALUE: 126 – PERCENTUAL: 25.0%

LABEL w – VALUE: 297 – PERCENTUAL: 58.928574%

MODE OF TRANSPORT ID: 4

LABEL s – VALUE: 126 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 5

LABEL s – VALUE: 126 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 6

LABEL s – VALUE: 126 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 7

LABEL c – VALUE: 197 – PERCENTUAL: 19.543652%

LABEL s – VALUE: 646 – PERCENTUAL: 64.0873%

LABEL w – VALUE: 165 – PERCENTUAL: 16.369047%

MODE OF TRANSPORT ID: 8

LABEL s – VALUE: 126 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 9

LABEL c – VALUE: 126 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 10

LABEL s – VALUE: 126 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 11

LABEL c – VALUE: 50 – PERCENTUAL: 39.68254%

LABEL s – VALUE: 26 – PERCENTUAL: 20.63492%

LABEL w – VALUE: 50 – PERCENTUAL: 39.68254%

MODE OF TRANSPORT ID: 12

LABEL bc – VALUE: 805 – PERCENTUAL: 2.4200337%

LABEL r – VALUE: 519 – PERCENTUAL: 1.5602453%

LABEL c – VALUE: 6869 – PERCENTUAL: 20.649952%

LABEL s – VALUE: 19218 – PERCENTUAL: 57.77417%

LABEL w – VALUE: 5853 – PERCENTUAL: 17.595598%

MODE OF TRANSPORT ID: 13

LABEL s – VALUE: 864 – PERCENTUAL: 62.33766%

LABEL c – VALUE: 252 – PERCENTUAL: 18.181818%

LABEL w – VALUE: 270 – PERCENTUAL: 19.480518%

MODE OF TRANSPORT ID: 14

LABEL s – VALUE: 65 – PERCENTUAL: 51.587303%

LABEL w – VALUE: 61 – PERCENTUAL: 48.412697%

MODE OF TRANSPORT ID: 15

LABEL c – VALUE: 23 – PERCENTUAL: 18.25397%

LABEL s – VALUE: 10 – PERCENTUAL: 7.936508%

LABEL w – VALUE: 93 – PERCENTUAL: 73.809525%

MODE OF TRANSPORT ID: 16

LABEL s – VALUE: 126 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 17

LABEL c – VALUE: 108 – PERCENTUAL: 85.71429%
LABEL w – VALUE: 18 – PERCENTUAL: 14.285715%

MODE OF TRANSPORT ID: 18
LABEL s – VALUE: 126 – PERCENTUAL: 100.0%

MODE OF TRANSPORT ID: 19
LABEL c – VALUE: 854 – PERCENTUAL: 35.687424%
LABEL s – VALUE: 1189 – PERCENTUAL: 49.68659%
LABEL w – VALUE: 350 – PERCENTUAL: 14.625992%

MODE OF TRANSPORT ID: 20
LABEL s – VALUE: 126 – PERCENTUAL: 100.0%

Sorgente 5.8: Struttura dei cluster individuati per il dataset IH1 con il feature set otto

I risultati del processo applicato alla settimana di dati registrati mediante Lifetracker non soddisfano pienamente le aspettative. Si verifica, infatti, un fenomeno simile a quello che accade con il dataset Geolife. Viene individuato nella maggior parte dei casi il mezzo più frequente, che costituisce circa il sessanta per cento del dataset (come si può notare dal grafico riportante il numero di dati grezzi per ciascun mezzo) ed in misura minore due ulteriori mezzi. I due rimanenti, invece, non vengono mai riconosciuti. Tale fenomeno, dunque, è meno grave rispetto ai casi analizzati in precedenza ma comunque presente. Il valore di precisione e la struttura dei cluster individuati, infatti, rispecchiano quanto analizzato in questo paragrafo.

Indice analitico

A

accelerometro, 2, 3, 8, 16, 24, 25, 35, 36, 39, 93
Affinity Propagation, 18, 70, 71, 80, 81
alberi di decisione, 41–44
almanacco, 13
antecedente, 41
AP, 72, 74, 80
apprendimento di rinforzo, 4
apprendimento non supervisionato, 3, 4, 17, 65, 66, 75, 93
apprendimento supervisionato, 3, 4, 17, 18, 33, 93
approccio dividi-e-conquista, 43, 44
ARFF, 38, 42
attributo, 33, 34, 41, 43, 44

B

Backpropagation, 52
Bayes Net, 48
bytecode, 14

C

C4.5, 44
campionamento di Gibbs, 78
centroide, 81

Chinese Restaurant Franchise, 79
Chinese Restaurant Process, 76
classificatore, 24, 41–45, 48–50, 52, 54
classificazione, 3, 14, 26, 33, 40, 93
Classify, 40
clustering, 3, 14, 17, 18, 24, 26, 65, 66, 75, 80, 93
conseguente, 41
CRF, 80
CRP, 76, 78–80
CSS, 15

D

DAG, 48
data mining, 2, 5, 6
dataset, 23, 34, 46, 93
dati effemeridi, 13
dimensionalità, 93
Dirichlet Process Mixture Model, 67, 75
disponibilità, 81
distribuzione di Bernoulli, 48
distribuzione di Gauss, 48, 76
distribuzione di probabilità, 4, 6, 44, 49
distribuzione multinomiale, 48
documento, 66
DPMM, 16, 18, 67, 75, 79, 80

E

entropia, 44
esemplare, 80
Experimenter, 38
Explorer, 38, 40

F

feature, 16, 33, 34, 37, 45
feature extraction, 4, 34
feature phone, 1
feature selection, 4
feature vector, 34, 37, 38, 50, 51,
55, 67, 78, 79
feed-forward, 51
fenomeno aleatorio, 6
finestra scorrevole, 67, 69
folds, 41, 45
funzione di densità, 4, 7
funzione di ripartizione, 7

G

Gaussian Dirichlet Process
Mixture Model, 67, 75
GDPM, 75, 79
Geolife, 26, 83, 84, 93, 97
gimbals, 9
giroscopio, 2, 3, 9, 24, 25, 35, 36,
93
GPS, 2, 3, 12, 14, 16, 17, 24, 25,
39, 93
GUI, 74

H

haversine, 37
HDP, 16, 69, 79, 83

Hierarchical Dirichlet Process, 69,
79
Hierarchical Polya-Urn Scheme, 80
HTML, 14

I

information gain, 44

J

J48, 44, 56
Java, 14, 75
Java FX, 15
Javadoc, 14
JDK, 15
JSP, 95
JVM, 14

K

K-Means, 76
k-medoids, 81
Knowledge Flow, 38

L

LDA, 16, 76
Lifetracker, 23, 55, 82–84, 94

M

macchine a vettori di supporto, 52
machine learning, 2, 3, 6, 18, 33,
48, 55, 93
magnetometro, 2, 3, 10, 24, 25, 35,
36, 93
magnitudo, 25, 81

Maximum-Margin Hyperplane, 53
 MDPMM, 75, 79
 mixture component, 80
 mixture model, 80
 MultilayerPerceptron, 50
 multimedia phone, 1
 Multinomial Dirichlet Process
 Mixture Model, 67, 75

N

NaiveBayes, 46, 48
 neurone, 50

O

OneRule, 42, 43, 45
 Overfitting, 40, 68

P

parametro di dispersione, 78
 parola artificiale, 68, 69
 parser, 26
 pattern, 5
 Perceptron, 50
 phablet, 1
 Polya-Urn Model, 79
 probabilità, 6, 46
 processo di Dirichlet, 78, 79
 pruning, 42, 43

R

RandomForest, 44
 regole, 41
 regole di classificazione, 41, 42
 regressione, 4
 responsabilità, 81

reti neurali, 50, 52
 RIA, 15
 riduzione della dimensione, 4, 16,
 34

S

Sample CLI, 38
 similarità, 81
 smartphone, 1, 23
 sovrapposizione, 67, 69
 spazio vettoriale, 10, 52
 splitting, 24
 Stick-Breaking Process, 79
 stima della densità, 4
 supporto, 7
 SVM, 52

T

tablet, 1
 teorema di Bayes, 6, 45–47
 topic, 66
 topic mixture, 66, 69, 70, 80, 82
 topic model, 66
 training set, 33, 40, 41, 43–45, 48,
 49, 52
 Twente, 26, 83, 84, 93

V

variabile casuale, 7
 vettore di supporto, 52–54

W

Weka, 18, 37, 38, 41, 49, 53, 54, 93
 words, 66

Bibliografia

- [1] I. Witten, E. Frank, M. Hall, *Data Mining: Practical Machine Learning Tools and Techniques* (2011), Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann Pub.
- [2] M. Lin, W. Hsu, *Mining GPS data for mobility patterns: a Survey*. Pervasive and Mobile Computing (2014), Vol. 12, pp 1–16, <http://elsevier.com/locate/pmc>
- [3] F. Sun, Y. Yeh, H. Cheng, C. Kuo, M. Griss, *Nonparametric Discovery of Human Routines from Sensor Data* (2014), IEEE International Conference on Pervasive Computing and Communications, Department of Computer Science, Stanford University.
- [4] J. Kwapisz, G. Weiss, S. Moore, *Activity Recognition using Cell Phone Accelerometers* (2010), ACM SIGKDD Explorations Newsletter, vol. 2, pp. 74–82, Department of Computer and Information Science, Fordham University.
- [5] A. Rasekh, C. Chen, Y. Lu, *Human Activity Recognition using Smartphone* (2011), Fall CSCE666 Project Report, Texas AM University.
- [6] B. J. Frey, D. Dueck, *Clustering by passing messages between data points* (2007), Science, vol. 315, pp. 972–976, <http://www.sciencemag.org/content/315/5814/972.full>
- [7] V. Vrinotys, *Overview of Cluster Analysis and Dirichlet Process Mixture Models* (2014), Machine Learning & Statistics, <http://blog.datumbox.com/overview-of-cluster-analysis-and-dirichlet-process-mixture-models/>
- [8] V. Vrinotys, *Finite Mixture Model based on Dirichlet Distribution* (2014), Machine Learning & Statistics, <http://blog.datumbox.com/finite-mixture-model-based-on-dirichlet-distribution/>

- [9] V. Vrinotys, *The Dirichlet Process the Chinese Restaurant Process and other representations* (2014), Machine Learning & Statistics, <http://blog.datumbox.com/the-dirichlet-process-the-chinese-restaurant-process-and-other-representations/>
- [10] V. Vrinotys, *The Dirichlet Process Mixture Model* (2014), Machine Learning & Statistics, <http://blog.datumbox.com/the-dirichlet-process-mixture-model/>
- [11] V. Vrinotys, *Clustering documents and gaussian data with Dirichlet Process Mixture Models* (2014), Machine Learning & Statistics, <http://blog.datumbox.com/clustering-documents-and-gaussian-data-with-dirichlet-process-mixture-models/>
- [12] V. Vrinotys, *Clustering with Dirichlet Process Mixture Model in Java* (2014), Machine Learning & Statistics, <http://blog.datumbox.com/clustering-with-dirichlet-process-mixture-model-in-java/>
- [13] E. Chen, *Infinite Mixture Models with Nonparametric Bayes and the Dirichlet Process* (2012), <http://blog.echen.me/2012/03/20/infinite-mixture-models-with-nonparametric-bayes-and-the-dirichlet-process/>
- [14] E. Chen, *Introduction to Latent Dirichlet Allocation* (2011), <http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/>
- [15] Nielsen Company, *The Mobile Consumer: a Global Snapshot* (2013), <http://www.nielsen.com/content/dam/corporate/us/en/reports-downloads/2013%20Reports/Mobile-Consumer-Report-2013.pdf>
- [16] M. Pawlan, *What is JavaFX?* (2013), JavaFX Documentation, <https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm>
- [17] I. Fedortsova, *Integrating JavaFX into Swing applications* (2013), JavaFX Documentation, <https://docs.oracle.com/javafx/2/swing/swing-fx-interoperability.htm#CHDIEEJE>
- [18] J. Gordon, A. Kouznetsov, *Skinning JavaFX Applications with CSS* (2013), JavaFX Documentation, https://docs.oracle.com/javafx/2/css_tutorial/jfxpub-css_tutorial.htm
- [19] A. Redko, *Scatter Chart* (2014), JavaFX Documentation, <https://docs.oracle.com/javafx/2/charts/scatter-chart.htm>

- [20] A. Redko, *Line Chart* (2014), JavaFX Documentation, <https://docs.oracle.com/javafx/2/charts/line-chart.htm>
- [21] A. Redko, *Bar Chart* (2014), JavaFX Documentation, <https://docs.oracle.com/javafx/2/charts/bar-chart.htm>
- [22] Wikipedia, *K-Means Clustering*, http://en.wikipedia.org/wiki/K-means_clustering
- [23] Wikipedia, *Statistical Classification*, http://en.wikipedia.org/wiki/Statistical_classification
- [24] Wikipedia, *Cluster Analysis*, http://en.wikipedia.org/wiki/Cluster_analysis
- [25] Wikipedia, *Feature Extraction*, http://en.wikipedia.org/wiki/Feature_extraction
- [26] Wikipedia, *Unsupervised Learning*, http://en.wikipedia.org/wiki/Unsupervised_learning
- [27] Wikipedia, *Accelerometer*, <http://en.wikipedia.org/wiki/Accelerometer>
- [28] P. Vidoni, *Calcolo delle probabilità*, http://www.dies.uniud.it/tl_files/utenti/vidoni/Didattica/StatAppl/Lucidi_Stat_Appl_parte_2_feb14.pdf
- [29] Wikipedia, *Gyroscope*, <http://en.wikipedia.org/wiki/Gyroscope>
- [30] Wikipedia, *Magnetometer*, <http://en.wikipedia.org/wiki/Magnetometer>
- [31] Wikipedia, *Global Positioning System*, http://en.wikipedia.org/wiki/Global_Positioning_System
- [32] Y. Zheng, Q. Li, Y. Chen, X. Xie, W. Ma, *Understanding Mobility Based on GPS Data* (2008), ACM conference on Ubiquitous Computing, Seoul, Korea.
- [33] Wikipedia, *J48*, http://en.wikipedia.org/wiki/C4.5_algorithm
- [34] Wikipedia, *Bayesian Networks*, http://en.wikipedia.org/wiki/Bayesian_network
- [35] Bayes Nets, *Bayes Nets*, <http://bayesnets.com/>

- [36] Wikipedia, *Artificial Neural Network*, http://en.wikipedia.org/wiki/Artificial_neural_network
- [37] DTREG, *Multilayer Perceptron Neural Network*, <http://www.dtreg.com/mlfn.htm>
- [38] Wikipedia, *Support Vector Machine*, http://en.wikipedia.org/wiki/Support_vector_machine
- [39] Wikipedia, *Hierarchical Dirichlet Process*, http://en.wikipedia.org/wiki/Hierarchical_Dirichlet_process
- [40] Y. W. Teh, M. I. Jordan, M. J. Beal, D. M. Blei, *Sharing Clusters Among Related Groups: Hierarchical Dirichlet Processes* (2007), Computer Science Div., Department of Statistics, University of California at Berkeley, <http://www.stats.ox.ac.uk/~teh/research/npbayes/nips2004a.pdf>
- [41] Wikipedia, *Affinity Propagation*, http://en.wikipedia.org/wiki/Affinity_propagation
- [42] Microsoft Research, *Geolife GPS Trajectories* (2007-2012), <http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/>
- [43] University of Twente, *Pervasive System Research Datasets* (2014), <http://ps.cs.utwente.nl/Datasets.php>
- [44] M. Shoaib, H. Scholten, P. J. M. Havinga, *Towards Physical Activity Recognition Using Smartphone Sensors* (2013), Pervasive Systems, Department of Computer Science and Electrical Engineering, University of Twente, Enschede, The Netherlands.