

UNIVERSITÀ DI PISA

Dipartimento di Informatica
Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Dashboard per il progetto AInCP e analisi di dati cinematici per la diagnosi di emiparesi cerebrale

Relatori:
Prof. Giuseppe Prencipe
Prof.ssa Alina Sîrbu
Dr. Samuele Bonini

Controrelatrice:
Prof.ssa Anna Bernasconi

Presentata da:
Lorenzo Amorelli

Anno Accademico 2022/2023

Indice

I INTRODUZIONE **1**

II DASHBOARD PER L'ACQUISIZIONE E LA CATALOGAZIONE DEI DATI **4**

1 Introduzione **5**

2 Progettazione e sviluppo di un prodotto software: concetti base **6**
2.1 Modello e Ciclo di vita del software 6
2.2 Analisi dei requisiti 7
2.2.1 Casi d'uso 8
2.3 Diagramma delle classi 8
2.4 Diagramma di distribuzione 8
2.5 Architettura del sistema 9
2.5.1 Model View Controller 9
2.5.2 Caratteristiche dei componenti e interazioni . . 9
2.6 Principi che migliorano la qualità del software: SOLID 10
2.7 Validazione e Verifica 10
2.7.1 Tipi di test 11
2.7.2 Criteri per la progettazione di test 13

3 Progettazione del database: concetti base **15**
3.1 Progettazione concettuale 15

3.2 Progettazione logica 16
3.3 Progettazione fisica 16

4 La piattaforma **17**
4.1 Analisi dei requisiti 17
4.2 Diagramma delle classi 22
4.3 Database 22
4.4 Verifica del software 24
4.4.1 Test del form di inserimento dati rispetto al CSRF 24
4.4.2 Test del popolamento del form di inserimento dati personali 26
4.5 Diagramma di distribuzione 28
4.6 Strumenti utilizzati 28
4.6.1 Scelta dello stack 28
4.6.2 Highcharts 29
4.6.3 Testing 30

5 Note conclusive **31**

III ANALISI DEI DATI **33**

6 Introduzione **34**

7 Machine Learning: concetti base **35**
7.1 Tipi di Machine Learning 35
7.1.1 Apprendimento supervisionato 35
7.1.2 Apprendimento non supervisionato 36
7.2 Sviluppo di un modello 36
7.2.1 Pre-Processing dei dati 36
7.2.2 Selezione e addestramento del modello 36
7.2.3 Validazione del modello 36
7.2.4 Test del modello 38
7.3 Algoritmi di apprendimento non supervisionato 39
7.3.1 Principal Component Analysis (PCA) 39
7.3.2 Clustering e K-means 40
7.4 Algoritmi di apprendimento supervisionato 42
7.4.1 K-nearest neighbors 42
7.4.2 Random Forest 43

8 Preprocessing dei dati **44**
8.1 Comprensione dei dati 44
8.1.1 Dati di laboratorio 44
8.1.2 Dati a casa 45
8.1.3 Aggregazione dei dati 45
8.1.4 Bilanciamento e dimensione del dataset 45
8.2 PCA 46
8.3 Scaling dei dati 47

9 Analisi **49**
9.1 Allenamento di classificatori 49
9.1.1 K Nearest Neighbors 49
9.1.2 Random Forest 50
9.2 Clustering 52
9.2.1 Classificatori su cluster 52
9.2.2 Rappresentazione grafica dei cluster 54

IV Conclusioni **56**

10 Sviluppi futuri **57**

11 Commenti personali **59**

Parte I

INTRODUZIONE

Scopo dello studio

Lo scopo di questo studio è l'implementazione di una pipeline di lavoro a supporto dei ricercatori del progetto AInCP (clinical validation of Artificial Intelligence for providing a personalized motor clinical profile assessment and rehabilitation of upper limb in children with unilateral Cerebral Palsy), in un contesto di digital health. Questo progetto di ricerca, coordinato dall'Università di Pisa, mira a fornire strumenti di supporto alle decisioni cliniche basati su prove di evidenza per la diagnosi funzionale, la disamina dell' arto superiore e l'intervento a domicilio nei bambini con paralisi cerebrale unilaterale (UCP), sviluppando, testando e convalidando strategie di intelligenza artificiale economicamente vantaggiose.
L'approccio AInCP verte a:

- stabilire la diagnosi clinica e fornire una prognosi accurata per la risposta al trattamento dei singoli profili di UCP, utilizzando un approccio multimodale che comprende la fenotipizzazione clinica [1], l'imaging cerebrale avanzato e il monitoraggio in tempo reale del funzionamento dell'arto superiore;

- fornire un trattamento personalizzato e specifico a domicilio, sfruttando le tecnologie informatiche e l'intelligenza artificiale.

I soggetti cui verrà fatto riferimento in questo studio sono bambini con *Paralisi Cerebrale Infantile* (PCI), la più comune forma di disabilità fisica dell'infanzia. Per PCI si intende un disturbo del neurosviluppo che che interessa 2-3/1000 nascite e 40-100/1000 nascite pretermine e premature [2] e che si manifesta come un disturbo del movimento e della postura, che causa una limitazione dell'attività, dovuto a disturbi verificatisi nello sviluppo del cervello del feto o del bambino[4]. Esistono diverse forme di paralisi cerebrale, i soggetti in esame hanno, per la precisione, paralisi cerebrale **unilaterale**, ovvero una disabilità che causa una debolezza o un'immobilità su un lato del corpo. Per questo motivo si studia e si valuta l'asimmetria presente nei movimenti dei bambini, esaminando quanto viene utilizzato un arto (quello dominante) rispetto all'altro (quello affetto).

[1] Processo medico che mira a definire l'insieme di tratti osservabili e misurabili di un individuo. Viene utilizzata per descrivere e classificare le malattie genetiche e le sindromi, che spesso presentano un insieme distintivo di segni e sintomi.

Valutazione dell'arto superiore nell'ambito clinico

Per la diagnosi di questa patologia si fa, all'interno di questo studio, riferimento ad una scala chiamata Assisting Hand Assessment (AHA). Questa serve a valutare la capacità di utilizzo dell'arto affetto durante attività bimanuali. La scala varia da un punteggio minimo di 0 (arto affetto non utilizzato), ad un punteggio massimo di 100 (nessuna asimmetria rilevata) [3]. Questa scala clinica deve essere somministrata e valutata da specialisti certificati e consta di attività strutturate, della durata di circa 20 minuti, diverse in base all'età del soggetto preso in esame.

Poichè la valutazione da parte del clinico è molto spesso qualitativa e dipendente dall'operatore, si è visto necessario quantificare in modo oggettivo il grado di asimmetria di un soggetto. Per questo scopo, un precedente studio [2] ha già appurato che, in un ambiente standardizzato, utilizzando sensori accelerometrici posizionati alle estremità di entrambi gli arti superiori è possibile determinare il maggior uso dell'arto più affetto rispetto a quello meno affetto. Questo sistema restituisce un valore oggettivo, che misura l'asimmetria nei movimenti.

Pipeline di lavoro ed obiettivi

Un primo ostacolo che è possibile trovare quando si parla di intelligenza artificiale è quello dell'acquisizione, del salvataggio e della catalogazione dei dati. Per questo motivo lo studio che viene presentato nella prima parte di questo lavoro intende porre le basi di una piattaforma utilizzabile dai ricercatori per risolvere questi problemi.

In secondo luogo, si è passati ad un lavoro di analisi utilizzando degli algoritmi di machine learning per studiare le attività dei soggetti partendo dai dati e dare supporto alle valutazioni dei medici, stabilite in base all'evidenza della condizione dei soggetti.

Parte II

DASHBOARD PER L'ACQUISIZIONE E LA CATALOGAZIONE DEI DATI

1. Introduzione

Non ci si prefigge l'arduo compito di consegnare un prodotto completo alla fine del lavoro di tesi, ma i ricercatori potranno beneficiare almeno delle funzionalità essenziali da loro richieste.
La descrizione e la definizione del problema da affrontare fanno parte di una fase dello sviluppo software. Non avendo intenzione di bruciare le tappe, questa non verrà esposta in questo primo capitolo introduttivo, dove viene proposta, invece, una considerazione.
Nello svolgimento di questi tipi di attività si è naturalmente portati a voler ottenere dei risultati il prima possibile, spinti dal committente o alla ricerca di appagamento, incuranti del fatto che un' inavvertenza nelle prime fasi del ciclo di vita di un progetto possa rivelarsi, nel futuro, come un macigno che blocca la via e vanifica la fatica investita. Ma come il committente della costruzione di un edificio non si aspetta di vederlo comparire pochi giorni dopo l'accordo nonostante l'inizio subitaneo dei lavori, così la richiesta di un prodotto software su misura, articolato o meno, non può (e non dovrebbe) essere ultimata in modo fugace.
È importante ricordarsi di essere forniti di una dote che Leopardi definiva come *"la più eroica delle virtù"*: lasciarsi accompagnare dalla pazienza è infatti fondamentale nello svolgimento di tali compiti e porta, nel medio-lungo periodo, a risvolti positivi e duraturi.
Gutta cavat lapidem, non vi sed saepe cadendo.

2. Progettazione e sviluppo di un prodotto software: concetti base

In questo capitolo vengono introdotti alcuni concetti di base generali relativi alla progettazione e lo sviluppo di un prodotto software.
Prima di inoltrarsi nel vivo, un team di lavoro ha necessità di pianificare gli incarichi da svolgere. Esistono diversi modelli di sviluppo, ognuno suddivide le attività necessarie al processo, le ordina e definisce i criteri per i passaggi alle fasi successive. La scelta di un modello rispetto ad un altro avviene contemporaneamente alla prima attività di analisi del problema.

2.1 Modello e Ciclo di vita del software

Per lo sviluppo del prodotto si è scelto di seguire un modello di sviluppo incrementale, ovvero un metodo che si basa su un ciclo di feedback continuo tra il "team di sviluppo" e il committente e che prevede l'elaborazione del progetto in fasi successive, ognuna delle quali aggiunge una nuova funzionalità al sistema. Il prodotto finale viene decomposto in componenti, ognuno dei quali viene progettato e sviluppato separatamente. Alla fine di ogni ciclo una nuova build (componente) viene rilasciata. L'uso di questo modello favorisce la creazione di prototipi e dunque la validazione dei requisiti attraverso la comunicazione continua con il committente.
La scelta del modello di sviluppo descritto fa in modo che non sia preclusa la possibilità di ripartire dalla versione del prodotto precedentemente raggiunta per lo sviluppo del prodotto finale (il quale potrà anche avere nuovi

requisiti, evoluti, rispetto a quelli iniziali).
Di seguito verranno descritte le fasi relative alla progettazione e lo sviluppo della piattaforma: analisi dei requisiti, progettazione, implementazione, verifica e validazione e manutenzione. Nel modello incrementale questi passi sono eseguiti più volte in ciclo (figura 2.1), come descritto sopra.

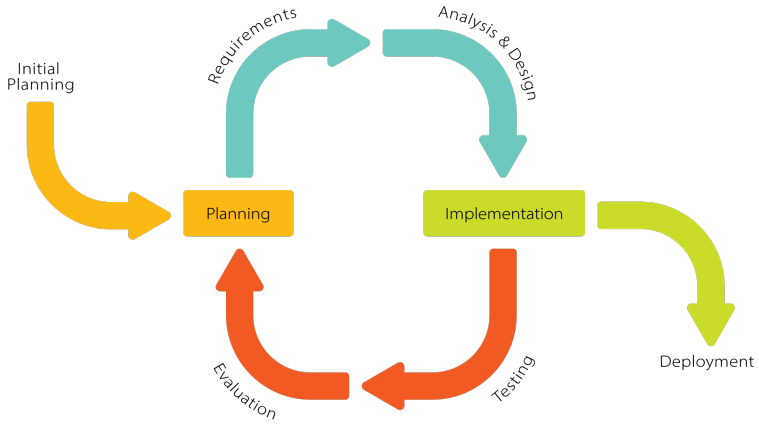


Figura 2.1: Il processo incrementale

Nelle sezioni seguenti vengono presentati i concetti di base relativi alle attività svolte, ai diagrammi e agli strumenti utilizzati. Nel prossimo capitolo, invece, i diagrammi e gli strumenti sono presentati nel merito del lavoro svolto.

2.2 Analisi dei requisiti

L'attività di analisi dei requisiti parte da una valutazione dei costi e dei benefici della realizzazione di un prodotto per poi passare alla definizione del problema da risolvere. Questa attività produce solitamente un modello (un'astrazione del sistema usata per specificarne struttura e comportamento) o un documento relativo alla descrizione del dominio, ovvero del campo di applicazione del prodotto, e dei requisiti, ovvero di ciò che deve essere garantito dal sistema per soddisfare le necessità di un utente.
I requisiti si dividono in due categorie:

- Funzionali: descrivono le funzionalità che il sistema deve prevedere per soddisfare le esigenze degli utenti

- Di qualità: descrivono le proprietà del sistema in relazione a determinate funzioni

Come supporto alla descrizione e progettazione del sistema è stato utilizzato il linguaggio di modellazione UML, attraverso il software Visual Paradigm.

2.2.1 Casi d'uso

I casi d'uso descrivono l'utilizzo che gli utenti (persone o cose) fanno del sistema. Il diagramma dei casi d'uso viene utilizzato per modellarne i requisiti funzionali. È composto da quattro entità:

- Attori: i ruoli assunti dagli utenti che usano il sistema
- Casi d'uso: i compiti che gli attori eseguono con l'ausilio del sistema
- Relazioni: le interazioni tra gli attori e i casi d'uso
- Confine del sistema: la linea di delimitazione che ingloba i casi d'uso. Rappresenta il confine del sistema oggetto del modello

In aggiunta alla modellazione descritta, per ogni caso d'uso è stata redatta una **descrizione narrativa**: un documento che descrive i diversi scenari che si possono presentare nell'esecuzione di un caso d'uso.

2.3 Diagramma delle classi

Il diagramma delle classi è la descrizione statica della struttura del dominio. Descrive i tipi degli oggetti che fanno parte di un sistema software e le relazioni tra essi che non cambiano nel tempo. Inoltre descrive proprietà e operazioni delle classi.

Una prima analisi dei requisiti del sistema ha reso possibile l'identificazione delle entità, delle relazioni tra esse e la loro rappresentazione nel diagramma, come classi.

2.4 Diagramma di distribuzione

Il diagramma di distribuzione (o deployment diagram) modella la distribuzione delle informazioni fisiche che vengono prodotte da un processo di sviluppo software (artefatti) sulle risorse fisiche sulle quali queste possono essere dispiegate (nodi). I nodi sono visualizzati come scatole e gli artefatti

assegnati a ciascun nodo come rettangoli all'interno delle scatole. Un singolo nodo in un diagramma di distribuzione può rappresentare concettualmente più nodi fisici, come un cluster di server di database.

2.5 Architettura del sistema

Lo stile architetturale selezionato per il progetto in questione prende il nome di Model-View-Controller (MVC).

2.5.1 Model View Controller

MVC è un pattern architetturale diffuso nell'ambito della programmazione orientata agli oggetti e non solo. È in grado di disaccoppiare la rappresentazione del modello di dominio, l'interfaccia utente e il controllo dell'interazione uomo-macchina. Si individuano tre componenti:

- Model: addetto alla gestione dei dati, si divide spesso in più classi e fornisce i metodi per accedere ai dati dell'applicazione;
- View: presenta all'utente una vista dei dati gestiti dal modello;
- Controller: si può considerare logicamente diviso in due parti. Una raggruppa le procedure che possono essere effettuate sul modello e l'altra comprende la gestione delle azioni sulla vista.

2.5.2 Caratteristiche dei componenti e interazioni

Per semplicità si suppone che i componenti siano rappresentati, ciascuno, da una sola classe. Il flusso di lavoro è descrivibile in 4 passi:

1. L'utente effettua un'azione sulla View (es. click su un link)
2. La gestione dell'azione è demandata al Controller attraverso un sistema di *routing* che lega l'azione richiesta al relativo metodo del controller.
3. Il Controller elabora (ottiene/modifica) i dati del Modello.
4. Il Controller procede all'aggiornamento della view che è completamente stateless e si limita semplicemente a fare rendering.

2.6 Principi che migliorano la qualità del software: SOLID

Un software di qualità, oltre ad essere affidabile, sicuro e prestante, è anche facilmente mantenibile, scalabile e testabile.

Con SOLID si intendono cinque principi di progettazione definiti per aiutare gli sviluppatori a creare software di qualità. I principi sono:

- Single Responsibility Principle: ad ogni classe si assegna una sola responsabilità. Una classe dovrebbe avere solo un motivo per cambiare, ovvero essere funzionalmente coesa.
- Open/Closed Principle: le classi dovrebbero essere aperte all'estensione ma chiuse alla modifica.
- Liskov Substitution Principle: gli oggetti di una sottoclasse dovrebbero essere utilizzabili ovunque venga utilizzato l'oggetto della classe padre.
- Interface Segregation Principle: si preferiscono molte interfacce specializzate ad una unica interfaccia generale.
- Dependency Inversion Principle: le classi astratte dovrebbero dipendere dalle classi concrete e non il contrario.

Nell'implementazione del sistema si è cercato di seguire i principi appena presentati in modo da ottenere un software scalabile, adattabile a nuove esigenze future e facilmente testabile.

2.7 Validazione e Verifica

Una parte fondamentale del processo di sviluppo software è rappresentata dalla V&V, validazione e verifica. L'attività di verifica ha lo scopo di controllare che durante la progettazione e lo sviluppo non sia tralasciato alcun requisito, la validazione, invece, è un'attività di controllo mirata a sincerarsi che il prodotto sia adeguato per l'uso previsto.

L'attività di **verifica** è spesso un processo interno al team di lavoro. Può essere suddivisa in due parti:

- Verifica statica: ha come obiettivo l'analisi del codice sorgente senza ricorrere alla sua esecuzione. Questo tipo di verifica può basarsi su strumenti automatici o meno. Le tecniche più comuni sono chiamate

inspection e *walkthrough*, entrambe hanno lo scopo di trovare e segnalare degli eventuali difetti ma non di effettuare correzioni: i gruppi di revisione e di codifica dovrebbero infatti essere diversi per non avere influenze l'uno sull'altro. La walkthrough è una emulazione dell'esecuzione del codice mentre lo si legge, la inspection, invece, è una lettura del codice focalizzata su aspetti ben definiti (si fissano gli errori e si ricercano).

- Verifica dinamica (test): ha come obiettivo l'analisi del comportamento del software durante l'esecuzione e comprende diversi aspetti.
  - Progettazione dei test: è l'aspetto orientato alla selezione dell'insieme dei dati di ingresso da fornire al programma.
  - Ambiente di test: deve permettere l'esecuzione controllata del software in esame, sia esso un programma completo o una sua parte.
  - Analisi dei risultati: il confronto tra i dati attesi dall'esecuzione del test e i risultati ottenuti
  - Debugging: l'attività di eliminazione dei difetti trovati. La fase finale del test.

La **validazione** è un'attività normalmente prevista sul prodotto finito e spesso coinvolge il committente. Tuttavia, limitandosi ad aspetti particolari, è comunque possibile effettuare delle operazioni di validazione anche durante il processo di sviluppo.

2.7.1 Tipi di test

Elenchiamo adesso tre diversi tipi di test per evidenziarne le differenze e le relazioni. Anticipiamo che per trovare il giusto equilibrio tra tutti e tre i tipi di test, il miglior ausilio visivo da utilizzare è la piramide dei test (figura 2.2). La maggior parte dei test è costituita da test unitari (si veda Unit testing), alla base. Man mano che si sale, i test diventano più grandi, ma allo stesso tempo il numero di test (la larghezza della piramide) si riduce [12].

Unit testing

È l'attività di collaudo di singole unità di software, come può essere una funzione, un metodo o una classe. Uscendo, per un attimo, dal mondo

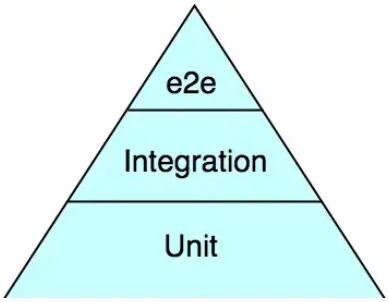


Figura 2.2: Versione semplificata della piramide di testing presentata da Google [1]

software e considerando le singole parole all'interno di un testo come delle unità, possiamo dire che l'attività di unit testing è effettuata, per esempio, dall'autocorrettore di un editor durante la stesura di un documento ufficiale, in quanto questo evidenzia le parole errate. Tornando al mondo software, lo unit testing presenta diversi limiti: si consideri ad esempio la situazione in cui il modulo da testare necessiti della comunicazione con altri moduli. Per testare questo tipo di unità è necessario costruire due tipologie di moduli:

- Moduli guida (*driver*): servono per invocare l'unità sotto test inviandole opportuni valori, relativi al test case. In linguaggi object-oriented, questi moduli devono costruire l'oggetto il cui metodo è sotto test e avviare il metodo da testare.

- Moduli fittizi (*stub*): sono moduli invocati dall'unità che stiamo testando. Emulano il funzionamento dei moduli chiamati da quello sotto test. Gli stub possono essere scritti sulla base di una conoscenza *black box* del modulo da emulare. Una volta che un modulo è stato realizzato e testato, può essere sostituito al suo relativo stub.

Per conoscere le dipendenze di un dato modulo da testare si ricorre all'utilizzo di un *grafo delle dipendenze*. In questo tipo di grafo i nodi rappresentano i moduli, mentre gli archi, orientati, rappresentano relazioni di dipendenza tra di essi. Questo grafo può essere ricavato dall'analisi del codice sorgente dell'applicazione e, in diversi casi, esistono dei tool automatici che sono in grado di costruirlo partendo appunto dai sorgenti.

Integration testing

Una volta testate le singole unità, il test di integrazione valuta se queste funzionino correttamente nel loro complesso. Si sostituiscono, nella pratica, gli stub con i loro rispettivi moduli. Esistono due strategie con le quali applicare l'integrazione. La strategia *bottom-up* consiste nel testare per primi i componenti di livello più basso. Questi ultimi vengono poi utilizzati per facilitare il test dei componenti di livello superiore (integrazione con moduli di livello via via più alto) e questo processo si ripete finché non viene testato il componente in cima alla gerarchia. La strategia *top-down*, invece consiste nel testare per primi i moduli ad alto livello e i rami del modulo vengono seguiti passo dopo passo (integrazione con moduli di livello via via più basso) fino alla fine del modulo corrispondente.

End-to-End testing

Terminata la fase di integration testing, si passa ad una fase volta a considerare il flusso di lavoro dell'utente finale, ponendo sotto test le funzionalità e le performance di un'applicazione in circostanze simili a quelle di un prodotto finale e con dati che replicano le impostazioni reali. L'obiettivo è quello di simulare dall'inizio alla fine lo scenario reale di un utente.

2.7.2 Criteri per la progettazione di test

Prima di concludere questa sezione, si presentano due criteri che guidano la progettazione dei test. Le caratteristiche proprie di queste due classi rendono conveniente scegliere i criteri da applicare in dipendenza della fase del processo di sviluppo in cui il prodotto è sottoposto a controllo [6].

Criteri funzionali

I criteri funzionali sono criteri basati su modelli a scatola chiusa (*black-box*, figura 2.3). Infatti non è importante *come* un modulo sotto test sia stato implementato ma *cosa realizza*. Questo tipo di criteri richiede l'analisi degli output generati dal sistema (o da suoi componenti) in risposta ad input (test cases) definiti sulla base della sola conoscenza dei requisiti del sistema (o di suoi componenti). Il punto comune di tutte le tecniche Black-Box è il fatto che il software è acceduto unicamente attraverso la sua interfaccia, senza accedere in maniera diretta al codice del componente da testare [11].



Figura 2.3: Esempio di black-box. Fonte: Wikipedia

Criteri strutturali

I criteri strutturali sono criteri che tengono conto della realizzazione del programma da controllare e quindi, a differenza dei criteri funzionali, del *come* sono realizzati. Sono detti *white-box* e sono basati sull'analisi del flusso di controllo di un programma. La progettazione dei test fa riferimento a un grafo di flusso (figura 2.4) ricavato dall'analisi del codice dove i nodi rappresentano un comando (statement) e gli archi orientati rappresentano il cammino, ovvero il legame di sequenzialità tra i comandi.

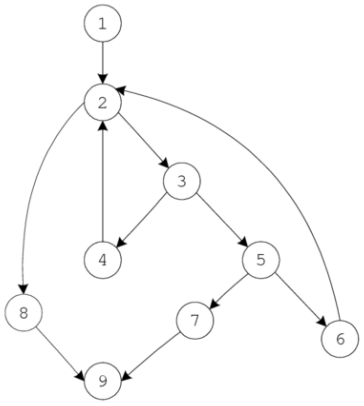


Figura 2.4: Esempio di grafo di flusso.

3. Progettazione del database: concetti base

La progettazione del database è il processo di formulazione di un modello del database. È possibile lavorarci dopo aver effettuato l'analisi dei requisiti e quindi dopo aver raccolto e studiato le funzionalità che il sistema dovrà avere. Generalmente si articola in tre fasi.

3.1 Progettazione concettuale

Ha lo scopo di rappresentare la realtà di interesse con una descrizione precisa e completa ma indipendente dai criteri di rappresentazione usati dal sistema informatico scelto per gestire la base di dati. Questa fase produce uno schema concettuale (figura 3.1).

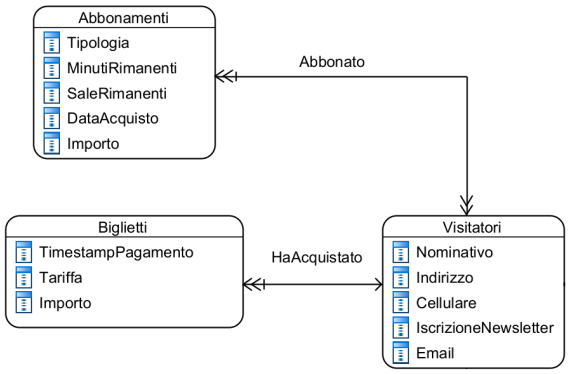


Figura 3.1: Esempio di diagramma concettuale.

3.2 Progettazione logica

L'obiettivo è tradurre il modello concettuale in un modello logico che sia compatibile con il DBMS scelto. In questa fase si definiscono le tabelle e le relazioni tra di esse, i campi, le chiavi primarie e vengono impostati i tipi di colonna. Produce uno schema logico che serve come base per la creazione fisica del database.

3.3 Progettazione fisica

Ha lo scopo di tradurre lo schema logico in termini di tabelle e relazioni che andranno a costituire la struttura fisica vera e propria del database. Rappresenta il modo in cui i dati dovrebbero essere strutturati e correlati in uno specifico DBMS. Ciò significa che è necessario un uso accurato del tipo di dati per le colonne delle entità e che bisogna evitare l'uso di parole riservate nella denominazione di entità e colonne. Inoltre, si aggiungono al progetto chiavi primarie, chiavi esterne e vincoli. Questa fase produce uno schema fisico (figura 3.2).

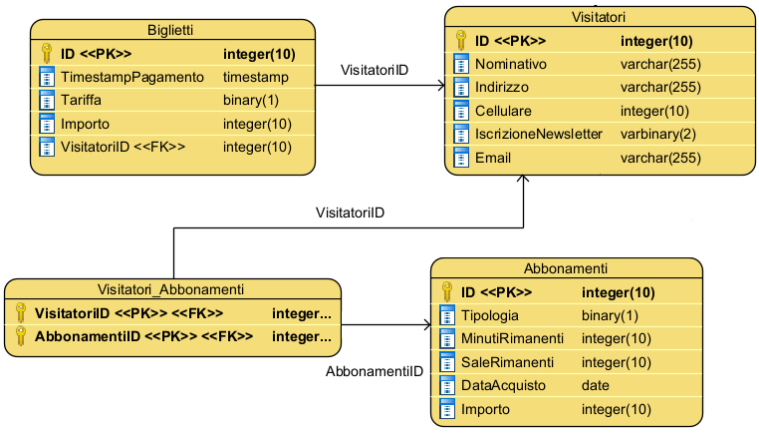


Figura 3.2: Esempio di diagramma fisico.

4. La piattaforma

4.1 Analisi dei requisiti

In questo capitolo si entra nel merito del lavoro di progettazione e sviluppo del software. Il primo passo è quello di studiare i requisiti.
Il contesto è quello nel quale i soggetti sono monitorati durante visite in ambulatorio (sessioni AHA) e durante lo svolgimento delle normali attività tra le mura di casa (sessioni week). Il monitoraggio avviene tramite l'utilizzo di due ActiGraph GT3X, ovvero dei dispositivi basati su sensori accelerometrici triassiali [7], posizionati alle estremità dei due arti superiori dei bambini. Questi permettono la registrazione e il salvataggio di dati ad una data frequenza e consentono, tramite l'utilizzo del software ActiLife, di esportarli e convertirli in un formato che verrà descritto più a fondo nel capitolo riguardante l'analisi dei dati. Per il momento basti sapere che questi dati sono esportati su file *.csv*. Ognuno di questi file contiene dati relativi a sessioni AHA e/o week: la necessità dei medici è quella di poter avere i dati salvati in un unico spazio e di poter etichettare delle parti di questi file attraverso un'interfaccia, in modo da disporre di un unico punto di accesso alle informazioni.

Si è scelto di sviluppare una web app, producendo così un software accessibile attraverso un browser web.

Diagramma dei casi d'uso

Il requisito essenziale per il committente è la possibilità di etichettare sottoinsiemi di dati partendo dai file che li contengono. Questo caso d'uso comprende l'eventuale inserimento dei metadati dei soggetti e l'autentica-

zione al sistema.
Altri casi d'uso sono rappresentati nel diagramma in figura 4.1.

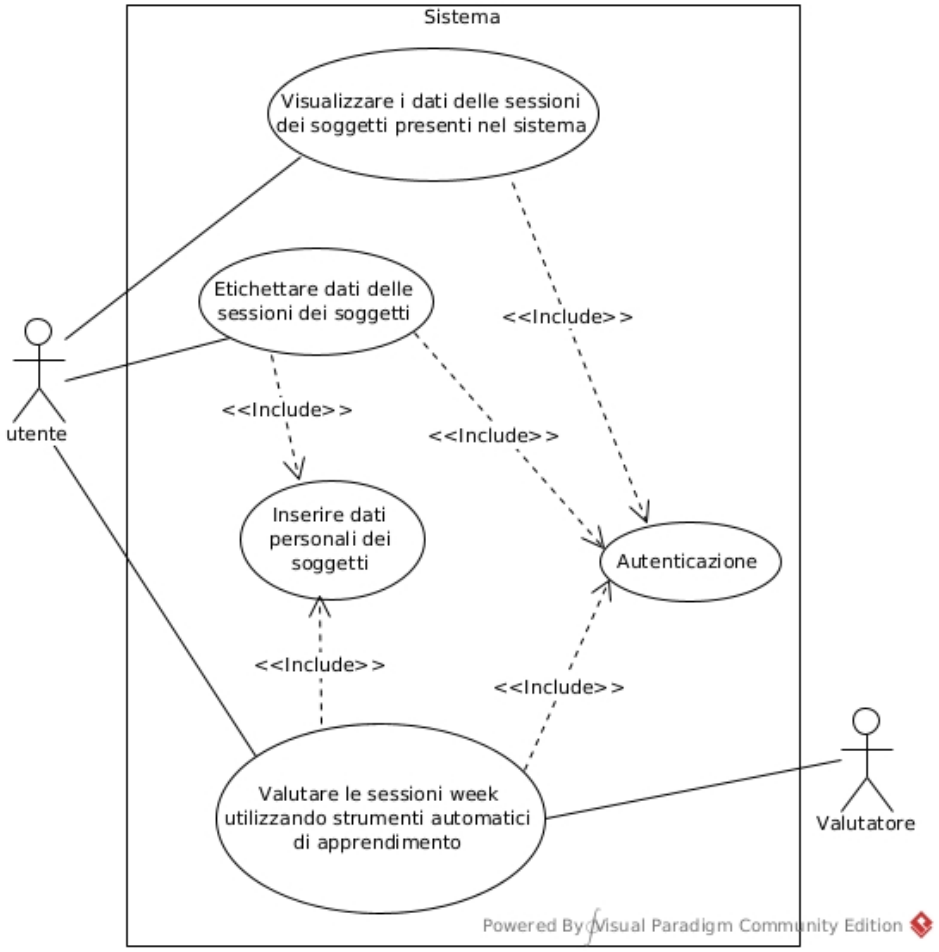


Figura 4.1: I casi d'uso in un diagramma

Descrizioni narrative dei casi d'uso

Di seguito vengono descritti i casi d'uso già implementati, presenti nel diagramma.

Autenticazione

È il caso d'uso incluso da tutti gli altri. L'accesso alla piattaforma è ristretto a coloro che conoscono le credenziali.

Breve descrizione:	L'utente vuole autenticarsi sulla piattaforma.
Attori primari:	Utente
Attori secondari:	-
Precondizioni:	L'utente ha aperto la piattaforma.
Eventi principali:	1. **if** utente ha già effettuato il login 1.1 utente è stato autenticato correttamente 1.2 **end** 2. utente inserisce user code e password 3. utente preme sul bottone "Sign In" apposito 4. **if** utente non ha inserito credenziali valide 4.1 utente è reindirizzato alla pagina di login, la stessa sulla quale ha inserito le credenziali 4.2 **alt end** 5. **else** 5.1 utente è stato autenticato correttamente
Postcondizioni:	L'utente ha accesso alle funzioni della piattaforma ed è stato reindirizzato alla dashboard.
Sequenze alternative:	L'utente non ha accesso alla piattaforma.

Inserire dati personali dei soggetti

Prima di poter lavorare con i dati delle sessioni è necessario aggiungere al sistema informazioni di base sui soggetti stessi, come il nome, il cognome e il codice soggetto. L'implementazione di questo caso d'uso consente l'inserimento dei metadati dei soggetti.

Breve descrizione:	L'utente vuole caricare sul server i dati di base di un soggetto.
Attori primari:	Utente
Attori secondari:	-
Precondizioni:	L'utente possiede i dati da caricare. Inoltre l'utente ha aperto la piattaforma.
Eventi principali:	1. **if** utente non ha già effettuato il login 1.1 **include** Autenticazione 1.2 **alt end 1** 2. utente seleziona la funzionalità di inserimento di un nuovo soggetto 3. utente inserisce i dati del soggetto richiesti 4. **if** dati inseriti non validi 4.1 l'inserimento dei dati non è andato a buon fine 4.2 **alt end 2** 5. utente salva le informazioni premendo sul bottone apposito
Postcondizioni:	L'utente ha effettuato il caricamento sul server dei dati di base di un nuovo soggetto, un nuovo record è presente nella tabella *subjects*.
Sequenze alternative:	**1**: L'utente non ha ancora accesso alla piattaforma. **2**: Un messaggio di errore è presente a video.

Etichettare dati delle sessioni dei soggetti

I dati delle sessioni sono i dati ottenuti dal software ActiLife, contenuti in file dal formato *csv*. Questi file possono contenere al loro interno sessioni week e/o sessioni AHA. Si vuole dare la possibilità ai ricercatori di selezionare una parte dell'insieme di dati contenuto in un file ed etichettarla.

Breve descrizione:	L'utente vuole etichettare dei dati delle sessioni dei soggetti. I dati, contenuti in un file, possono essere già presenti sul server oppure possono essere caricati dall'utente.
Attori primari:	Utente
Attori secondari:	-
Precondizioni:	L'utente possiede il file sul quale vuole lavorare oppure il file è presente sul server. Inoltre l'utente ha aperto la piattaforma.
Eventi principali:	1. **if** utente non ha già effettuato il login 1.1 **include** Autenticazione 1.2 **alt end 1** 2. **if** utente vuole lavorare su dati di un soggetto non presente nel sistema 2.1 **include** Inserire dati personali dei soggetti 2.2 **alt end 2** 3. utente apre pagina delle sessioni di uno dei soggetti presenti sul server 4. utente seleziona la funzionalità di inserimento di una nuova sessione 5. utente seleziona il file desiderato (da locale o da remoto dalla lista di file già presenti su server) 6. utente utilizza il navigatore sul grafico visualizzato per selezionare il range di dati da etichettare 7. utente sceglie l'etichetta desiderata, inserisce note opzionali e salva
Postcondizioni:	L'utente ha effettuato l'etichettatura di una porzione di dati, un nuovo record è presente all'interno della tabella *fileportions*. Se un nuovo file è stato caricato, un nuovo record è presente all'interno della tabella *filepaths*.
Sequenze alternative:	**1**: L'utente non ha ancora accesso alla piattaforma. **2**: L'utente è reindirizzato alla pagina di inserimento nuovo soggetto

4.2 Diagramma delle classi

Il diagramma in figura 4.2 è una vista statica del dominio. Si ha una classe Soggetto, con tutte le sue proprietà, che si relaziona con la classe Sessione nel seguente modo: un soggetto può essere protagonista di (o aver svolto) zero o più sessioni. La classe Sessione ha l'attributo *tag* che può prendere i valori AHA o WEEK.
La classe File è una composizione di Sessione in quanto, se un file viene eliminato dal sistema, le sessioni a questo legate non hanno più il riferimento e vengono eliminate a loro volta. Al contrario, un file può esistere ma non essere stata specificata ancora alcuna Sessione relativa ad esso.

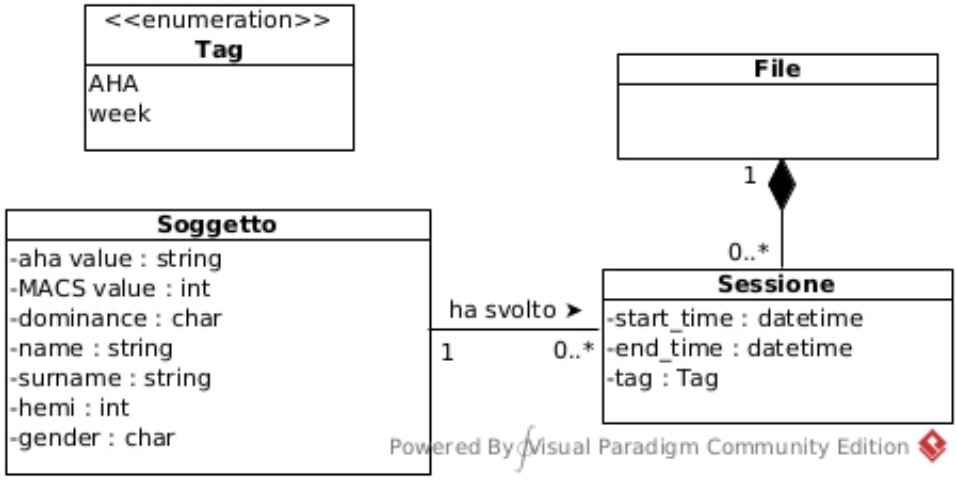


Figura 4.2: Il diagramma delle classi

4.3 Database

In questa sezione vengono presentati gli schemi concettuale e fisico, risultati della fase di progettazione del database. I soggetti vengono salvati all'interno della tabella SUBJECTS, i percorsi dei file presenti sul server all'interno della tabella FILEPATHS e i tag disponibili per l'etichettatura delle sessioni all'interno della tabella TAGS. Infine, la tabella FILEPORTIONS contiene i delimitatori, in formato datetime, che servono a determinare la sezione del file al quale si riferiscono, per la definizione di una sessione.

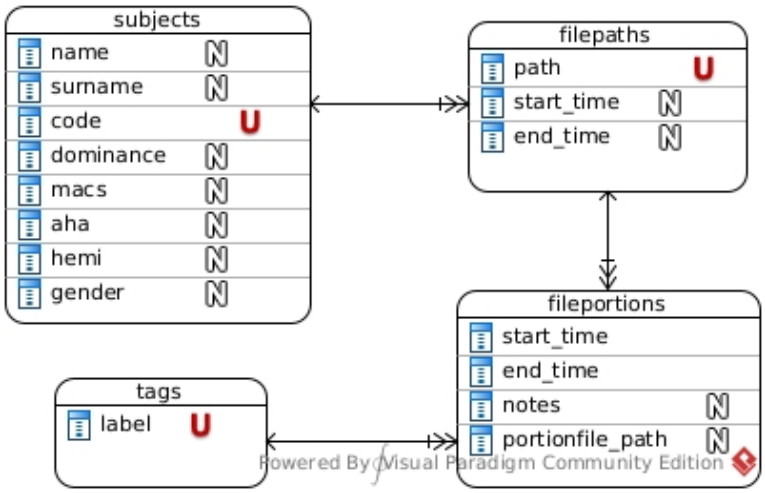


Figura 4.3: Il diagramma concettuale

Nello schema fisico si hanno i tipi dei campi delle tabelle e si possono notare le relazioni tra queste.

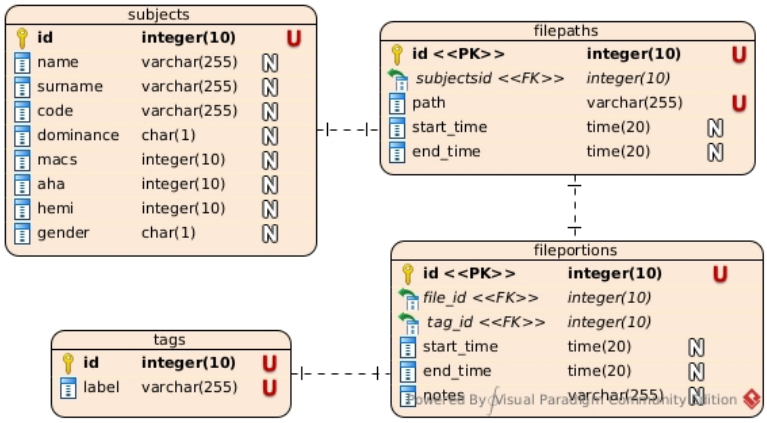


Figura 4.4: Il diagramma fisico

L'inserimento dei dati di un nuovo soggetto si traduce nell'inserimento di una nuova riga con i dati personali del soggetto all'interno della tabella SUBJECTS. La definizione di una nuova sessione, invece, si traduce nell'inserimento all'interno della tabella FILEPORTIONS di una nuova riga contenente i delimitatori della sessione, di tipo datetime, l'id di un record nella tabella TAGS, l'id di un record nella tabella FILEPATHS e le note lasciate dall'utente

relativamente alla sessione inserita. Da notare che il valore del campo chiave esterna *file_id* che punta alla tabella FILEPATHS potrebbe non esistere all'inizio della procedura di inserimento di una nuova sessione lato client, infatti l'utente è libero di scegliere se utilizzare un file già presente nel database o meno e, nel caso il file venga scelto tra quelli sulla macchina client (quindi non già presente nel sistema), la definizione della nuova sessione viene preceduta dal caricamento del file sul server e dall'inserimento di una nuova riga contenente il percorso di salvataggio dello stesso all'interno della tabella FILEPATHS.

4.4 Verifica del software

Dal momento che presentare i test di ogni modulo risulterebbe eccessivo e ridondante, vengono riportate solo le verifiche di un paio di funzionalità perché possano fare da esempio e modello anche per le altre, in quanto il modo di procedere è analogo.

4.4.1 Test del form di inserimento dati rispetto al CSRF

Nel primo test che viene presentato l'unità considerata è il form di inserimento dati delle sessioni degli utenti (figura 4.5). Il test non riguarderà il funzionamento o meno del form ma si concentrerà sulla prevenzione rispetto ad una possibile vulnerabilità, la CSRF.



Figura 4.5: Il form per l'inserimento dati sessione

La CSRF, ovvero la "falsificazione della richiesta di accesso al sito", è una debolezza a cui sono esposti i siti web dinamici quando sono progettati per ricevere richieste da un client senza meccanismi per controllare se la richiesta sia stata inviata intenzionalmente oppure no.
Si supponga, ad esempio, che un utente acceda alla piattaforma (es: aincp-platform.org), il server autentichi l'utente ed emetta una risposta che include un cookie di autenticazione. Ogni richiesta ricevuta da quel momento in avanti è una richiesta fidata se viene ricevuta con questo cookie di autenticazione in quanto il server ha la garanzia che sia stata inviata dal dispositivo dell'utente autenticato. Si supponga poi che l'utente visiti un sito malintenzionato che contiene un form simile al seguente:

```
<h1>Congratulations! You're the Winner!</h1>
<form action="https://aincp-platform.org/session/upload"
        method="post">
    <input type="hidden" name="start_time" value="1675182181"/>
    <input type="hidden" name="end_time" value="1675182202"/>
    <input type="hidden" name="file_id" value="10"/>
    <input type="hidden" name="notes" value="" />
    <input type="submit" value="Collect your prize!" />
</form>
```

Nel momento in cui l'utente seleziona il bottone, il browser esegue la richiesta includendo automaticamente il cookie di autenticazione per il dominio richiesto (aincp-platform.org). Il server dunque non può sapere che la richiesta non sia stata fatta intenzionalmente dall'utente e tratta l'operazione richiesta come lecita, provocando l'inserimento di una sessione.
L'esempio riportato non è distruttivo per il sistema in particolare ma azioni molto più pericolose possono essere eseguite seguendo lo stesso principio.

Per evitare questa vulnerabilità il sistema include nei form delle proprie pagine un token (token CSRF) composto da molte cifre casuali, diverso ogni per ogni pagina che viene servita. In fase di elaborazione della richiesta, dunque, il server si aspetta di trovare il token che era stato assegnato alla pagina e in caso contrario rifiuta la richiesta.

Il framework utilizzato per la fase di testing, presentato più avanti, permette di definire ed eseguire dei test sulle singole unità. Il codice proposto serve a sincerarsi che la protezione contro attacchi basati sulla CSRF sia attiva: il primo blocco è un test che, tramite il metodo `expectException` , si aspetta il lancio di un'eccezione `SecurityException`:

```
public function testCSRFBlock(){
    $this->expectException(
        \CodeIgniter\Security\Exceptions\SecurityException
            ::class);

    $this->call('post', 'session/upload', [
        'start_time'  => 1675182181,
        'end_time' => 1675182202,
        'file_id' => 10,
        'notes' => '',
    ]);
}
```

Il secondo invece si aspetta che non sia lanciata alcuna eccezione e che la risposta alla richiesta sia caricata correttamente. È possibile notare un campo ulteriore rispetto al test precedende, quello valorizzato col token CSRF di cui sopra.

```
public function testCSRFUpload(){
    $result = $this->call('post', 'session/upload', [
        Services::security()->getTokenName()
            => Services::security()->getHash(),
        'start_time'  => 1675182181,
        'end_time' => 1675182202,
        'file_id' => 10,
        'notes' => '',
    ]);

    $this->assertTrue($result->isOK());
}
```

4.4.2 Test del popolamento del form di inserimento dati personali

Un altro test interessante alla luce di quanto introdotto è il test sul popolamento del form di inserimento dati personali. Tramite questo modulo l'utente può inserire e modificare i dati personali dei bambini e nello specifico caso in cui si abbia la necessità di modificare i dati già presenti nel sistema, il form viene popolato automaticamente all'apertura con i dati del soggetto. Il test che viene presentato è utilizzato per testare il corretto popolamento.

Inizialmente, per il test dell'unità viene utilizzato uno **stub** che simula il recupero delle informazioni dal database:

```
private function stub_getSubjectDataFromDb(){
    $subjectStub = array();
    $subjectStub['id'] = '1';
    $subjectStub['name'] = "Mario";
    $subjectStub['surname'] = "Stub";
    $subjectStub['code'] = "123";
    $subjectStub['dominance'] = "R";
    $subjectStub['macs'] = '5';
    $subjectStub['aha'] = '80';
    $subjectStub['hemi'] = '1';
    $subjectStub['gender'] = "M";
    return $subjectStub;
}
```

Il test è dunque eseguito passando i dati del soggetto fittizi al metodo sotto test.

```
public function testSubjectCardPopulation(){
    // get subject data
    $subjectData = $this->stub_getSubjectDataFromDb();

    $result = $this
        ->controller(\App\Controllers\SubjectsController::class)
        ->execute('subject_card', $subjectData);

    $this->assertTrue($result->isOK());
    $result->assertSee('Subject Card', 'li');
    foreach ($subjectData as $key => $value) {
        $result->assertSee($value);
    }
}
```

Successivamente, una volta che il modulo per il recupero delle informazioni da db è completato e testato, lo stub viene rimosso. Questo passaggio si traduce nella sostituizione della riga

```
$subjectData = $this->stub_getSubjectDataFromDb();
```

con

```
$subjectData = $this->loadSubject($id);
```

dove il metodo `loadSubject` è definito come

```
protected function loadSubject($id){
    $subjectsModel = new SubjectModel();
    return $subjectsModel->where('id', $id)->first();
}
```

Quest'ultimo esempio è rappresentativo di un test di integrazione top-down: il modulo a livello più basso (che svolge il recupero dei dati) è inizialmente rimpiazzato da un modulo fittizio durante il test del modulo a livello più alto (il form). Quando anche il modulo di livello più basso è completato e testato, allora il nuovo modulo da testare si riduce all'unione dei due moduli.
È possibile notare, inoltre, che il modello `SubjectModel` in `loadSubject` è già stato verificato in precedenza in quanto parte di un modulo considerato verificato a sua volta.

4.5 Diagramma di distribuzione

Il diagramma di distribuzione in figura 4.6 descrive la dislocazione dei componenti software utilizzati. L'architettura è molto semplice, si hanno sia il server web col motore php sia il DBMS sullo stesso nodo.

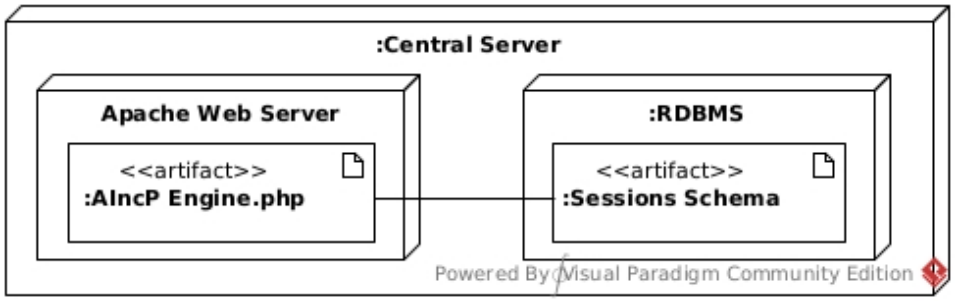


Figura 4.6: Il diagramma di distribuzione

4.6 Strumenti utilizzati

4.6.1 Scelta dello stack

La scelta del tech stack è stata piuttosto immediata data l'esperienza lavorativa pregressa del sottoscritto, si è dunque optato per una piattaforma LAMP, dalle iniziali dei componenti software utilizzati: LINUX, APACHE, MYSQL, PHP.

Il server web

Il server web utilizzato è **Apache 2.4**, ovvero la più recente versione stabile ad oggi. Il server web è il motore che fa da ponte tra gli script utilizzati per elaborare le richieste http e il punto di ingresso di queste ultime. In architetture più complesse è possibile trovare la macchina che agisce da server web come punto finale di una rete all'interno della quale i percorsi delle richieste sono gestiti da dei dispositivi chiamati *reverse proxy*. Questi ultimi si interfacciano direttamente con i client e possono aggiungere un livello di sicurezza ulteriore alla rete sia in termini di protezione contro minacce esterne sia in termini di stabilità ed efficienza, agendo da *load balancer*.

Database

Il DBMS utilizzato è MySQL. È un sistema di gestione di database relazionali composto di server. Un database relazionale è un tipo di database che memorizza e fornisce accesso a dati correlati tra loro. I dati sono salvati in tabelle e possono esistere relazioni tra dati ospitati in tabelle diverse. Un db relazionale si differenzia da uno non relazionale in quanto questo ultimo tipo di database non usa il modello relazionale nè lo schema tabulare per il salvataggio dei dati.

PHP Framework: CodeIgniter 4

CodeIgniter è un framework open-source per applicazioni web basato sul paradigma MVC, scritto in PHP. Ha una struttura compatta e veloce, ed è facilmente installabile. Nel 2008 parlando al frOSCon, Rasmus Lerdorf, uno dei creatori del linguaggio PHP, dichiarò che CodeIgniter fosse uno dei suoi framework preferiti proprio per queste caratteristiche.

Oltre a raggiungere performance eccellenti, il framework prende in seria considerazione la sicurezza offrendo protezioni integrate contro attacchi come CSFR, XSS e SQL injection.

La gestione degli utenti all'interno di CodeIgniter 4 viene effettuata attraverso un'estensione del framework chiamata *Shield*, che fornisce una serie di funzionalità per la sicurezza e la gestione di utenti con autenticazione e autorizzazione.

4.6.2 Highcharts

Highcharts è una libreria software per la creazione di grafici scritta in JavaScript. Viene utilizzata per la visualizzazione dell'indice di asimmetria

nel tempo (figura 4.7), in modo da concedere all'utente un feedback visivo durante la selezione del range in fase di inserimento di una nuova sessione. L'indice di asimmetria (AI) è una misura della differenza di movimenti tra i due arti ed è definita come

$$AI = \frac{vec_D - vec_{ND}}{vec_D + vec_{ND}} \cdot 100 \tag{4.1}$$

dove vec_D e vec_{ND} corrispondono rispettivamente alla magnitudo del vettore risultante dalla somma degli Activity Counts sui tre assi per il braccio dominante e non dominante. Questo indice è calcolato per ogni riga dei file *.csv* caricati dall'utente.

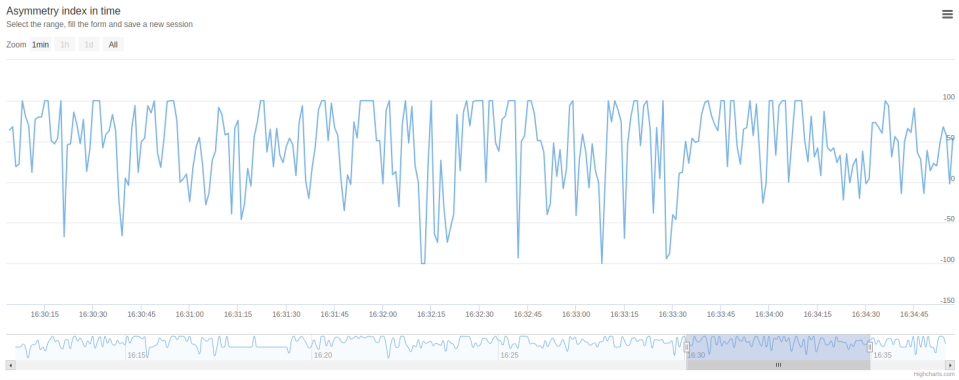


Figura 4.7: L'andamento di AI tra le 16:30 e le 16:35 in un file di esempio

4.6.3 Testing

PHPUnit è un framework di unit testing per il linguaggio PHP. Consente di testare piccole unità di codice e confrontare il risultato ottenuto rispetto ai risultati previsti, per individuare con facilità eventuali bug. Come altri framework di unit testing, PHPUnit utilizza le asserzioni per verificare che il comportamento del modulo testato si comporti come previsto. È dunque un framework adatto ad eseguire unit testing e integration testing.

5. Note conclusive

Si riportano brevemente alcune note relative alle fasi di vita del software in questione, con uno sguardo al modello incrementale.
Il lavoro è iniziato con un'**analisi dei requisiti**, producendo un diagramma che ha reso possibile la progettazione di funzionalità ben definite. Si è passati poi alla stesura di un diagramma per la **modellazione del problema** nel suo insieme, alla progettazione del **database** e si è optato per lo **stile architetturale** MVC. Si è dunque potuta confermare la scelta della tecnologia da utilizzare per lo sviluppo dell'applicazione web ed è iniziata l'**implementazione** delle prime funzioni di base del sistema. Una volta conclusa questa parte, il primo **prototipo** è stato reso accessibile al responsabile designato per provare il sistema ed è iniziata la fase di **test** e **valutazione** delle funzionalità implementate. In questa fase è stato individuato un problema prestazionale: la velocità dell'algoritmo di paginazione personalizzato che è stato implementato è inversamente proporzionale alla grandezza del file su cui lavora.
Per una maggiore comprensione del contesto si consideri che i file che sono caricati dall'utente possono essere molto grandi ($\sim$ 20MB): il calcolo dell'AI per ogni riga comporta, lato server, un costo computazionale eccessivo e, lato client, i punti da visualizzare sul grafico risultano troppi per la memoria di un elaboratore medio. È stato dunque necessario fissare il numero di punti del grafico da visualizzare e, dato il range di tempo considerato, effettuare un campionamento su di esso. La prima versione dell'algoritmo però, effettua due volte la scansione del file: la prima per recuperare gli indici di inizio e fine range dai delimitatori temporali e la seconda per costruire l'array di punti da visualizzare nel grafico.
Il core del corpo del metodo per la costruzione dell'array è il seguente:

```
$plot = array();
// init iterations values
$index = $start_index;
$step_size = ceil(($end_index - $start_index) /
    $max_number_of_rendered_points);
if($step_size<=0)
    $step_size=1;

while ($index < $end_index) {
    if($splFile->eof()){
        break;
    }
    $splFile->seek($index);
    $row = $splFile->fgetcsv();
    if(strcmp($row[0], 'datetime')==0){
        $index += $step_size;
        continue;
    }
    list($datetime, $x_D, $y_D, $z_D, $x_ND, $y_ND, $z_ND) =
        $row;
    $vec_D = $this->getVecMag($x_D, $y_D, $z_D);
    $vec_ND = $this->getVecMag($x_ND, $y_ND, $z_ND);
    $vecSum = $vec_D + $vec_ND;
    $AI = $vecSum==0 ? 0 : 100*($vec_D - $vec_ND)/($vecSum);
    $t = DateTimeImmutable::
        createFromFormat("Y-m-d H:i:s", $datetime);
    if($t)
        $plot[] = [$t->format('U')*1000, floor($AI)];
    $index += $step_size;
}
return $plot;
```

Si nota come si lavori con gli indici di inizio e fine range e si ha dunque la necessità di ottenerli in una fase precedente, scansionando il file fino al timestamp di fine intervallo.

Siamo dunque venuti a conoscenza che il requisito di prestazione, implicitamente richiesto da questo sistema, non è soddisfatto e ciò suggerisce un aspetto da considerare nella progettazione e implementazione del secondo prototipo. Grazie al modello incrementale, sarà possibile partire dalla solida base costruita durante tutte le fasi del primo ciclo di vita che, adesso, è possibile dichiarare concluso.

Parte III

ANALISI DEI DATI

6. Introduzione

La pipeline di lavoro di cui sopra prosegue con l'analisi dei dati. Abbiamo iniziato il percorso fornendo un modo accessibile, veloce e sicuro per salvare e catalogare i dati dei soggetti, che adesso sono dunque pronti per essere recuperati ed utilizzati dai ricercatori.

Il progetto AInCP mira alla validazione di nuovi algoritmi di intelligenza artificiale per la diagnosi funzionale e la tele-riabilitazione personalizzata di soggetti con emiplegia [10] e lo studio che andiamo a presentare ha come scopo la ricerca e la classificazione delle attività che vengono svolte dai bambini durante le sessioni a casa. È stato allenato un modello con la speranza di poter affiancare ai medici un sistema in grado di predire la valutazione AHA dei nuovi soggetti partendo dai dati delle sessioni WEEK, ovvero dai dati rilevati dagli attigrafi indossati dagli stessi durante la giornata, nell'arco di una settimana (un approfondimento di questi dati verrà presentato più avanti).

Il prossimo capitolo introdurrà le basi dell'apprendimento automatico e degli algoritmi utilizzati durante questo studio, poi verranno descritte le fasi di preprocessing e analisi dei dati nel merito del lavoro svolto e i risultati ottenuti.

7. Machine Learning: concetti base

In questo capitolo vengono introdotti alcuni concetti base relativi ai tipi di apprendimento automatico, alle fasi che portano allo sviluppo di un modello e agli algoritmi utilizzati per questo studio.

Il machine learning si basa sull'idea che sia possibile costruire modelli matematici in grado di fare previsioni o prendere decisioni basate su dati di esempio, anziché seguire istruzioni esplicite. Viene utilizzato in vari settori, come la diagnostica medica, il marketing e la finanza per fare previsioni e prendere decisioni più accurate rispetto a quelle basate su regole predefinite.

7.1 Tipi di Machine Learning

Esistono diversi tipi di apprendimento automatico che possono essere scelti in base all'obiettivo che si vuole raggiungere. Eccone alcuni tipi:

7.1.1 Apprendimento supervisionato

In questo tipo di machine learning il modello viene addestrato su un dataset etichettato, ovvero con dati di input e le rispettive uscite desiderate (etichette). Il modello impara a fare previsioni basate sui dati in ingresso durante la fase di allenamento e può essere utilizzato per fare previsioni su dati nuovi mai visti prima.

Gli algoritmi di apprendimento supervisionato si dividono in due categorie: classificazione e regressione. La regressione è utilizzata per prevedere una variabile continua, o meglio una funzione generalmente reale di variabili reali $f : \mathbb{R}^n \to \mathbb{R}^n$. La classificazione è utilizzata invece per prevedere una

variabile categorica o funzione $f : \mathbb{R}^n \to A$ in un insieme con cardinalità finita.

7.1.2 Apprendimento non supervisionato

In questo tipo di apprendimento il modello viene addestrato su un dataset non etichettato, ovvero con i soli dati di input, senza le rispettive uscite. Il modello cerca di trovare pattern o relazioni nei dati per raggrupparli in categorie. Un esempio di questo tipo di machine learning è il clustering.

7.2 Sviluppo di un modello

Successivamente alla definizione del problema, comprensione di come il machine learning può contribuire a risolverlo e alla raccolta dei dati si passa alle seguenti fasi:

7.2.1 Pre-Processing dei dati

Questa fase si pone l'obiettivo di verificare che i dati siano strutturati in modo adeguato per l'addestramento del modello. Si possono rimuovere valori duplicati, standardizzare il dataset o codificare le etichette.

7.2.2 Selezione e addestramento del modello

In base al tipo di problema che si vuole risolvere, è necessario scegliere l'algoritmo di machine learning più adeguato. Una volta effettuata questa scelta, si passa all'addestramento sui dati raccolti. È in questa fase che si fissano gli iperparametri del modello, ovvero delle proprietà che possono influire sulla complessità di esso e sulle sue prestazioni. Se si tratta di apprendimento supervisionato, il modello esposto ai dati impara a fare previsioni basandosi su di essi.

7.2.3 Validazione del modello

In questa fase si valutano le prestazioni del modello allenato. Esistono varie metriche per questo tipo di valutazione, la scelta ricade su quelle più appropriate per il problema. Di seguito, alcune delle metriche utili in questa fase:

- **Precisione** : è la percentuale di predizioni corrette rispetto al totale di predizioni fornite. Siano TP le predizioni corrette e FP le predizioni

errate, la precisione è definita come

$$P = \frac{TP}{TP + FP} \tag{7.1}$$

- **Recall**: è il rapporto tra il numero di predizioni corrette e il numero di elementi effettivamente appartenenti alla classe. Sia FN il numero di elementi fissata una classe, il recall è definito come

$$R = \frac{TP}{TP + FN} \tag{7.2}$$

- **Punteggio F1**: combina precisione e recall in un unico valore. È la media armonica di precisione e recall.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{7.3}$$

Overfitting e Cross validation

La misurazione della buona riuscita di un allenamento è necessariamente da valutare sui dati a disposizione stessi. Tuttavia un modello che si limita a ripetere le etichette delle osservazioni appena viste avrebbe un punteggio perfetto, ma non riuscirebbe a prevedere risultati corretti su dati mai visti prima. Questa situazione è meglio conosciuta come **overfitting** e per evitarla si ricorre ad una divisione del set di dati in una parte dedita all'allenamento del modello (*training set*) ed una parte dedita invece a testarlo (*test set*).
Purtroppo, questa divisione non basta ad evitare del tutto il problema di overfitting del modello: nella fase di scelta degli iperparametri per il processo di apprendimento si è naturalmente portati a trovare quelli che rendono tendente allo zero gli errori sul test set. Questo è tendenzialmente errato in quanto c'è il rischio di overfitting su quest'ultimo e le prestazioni di generalizzazione risentirebbero della previa conoscenza di tale insieme. Per ovviare a questo problema si divide il training set ulteriormente estrapolandone una parte, questo nuovo insieme prende il nome di *validation set*.
Dunque, l'utilizzo dei tre insieme descritti è dunque il seguente: l'allenamento del modello è effettuato sul training set, la valutazione (ovvero la misura della buona riuscita di tale allenamento) è fatta sul validation set e, una volta che l'esperimento è considerato soddisfacente, la valutazione finale è effettuata sul test set.

Purtroppo, il partizionamento del dataset in tre differenti insiemi riduce il numero di osservazioni utilizzabili per l'apprendimento del modello. Nei casi in cui il dataset contiene pochi dati, questo è inaccettabile.
Una soluzione a questo problema è chiamata **Cross-Validation (CV)**. Questo metodo prevede la divisione del dataset in training set e in test set. Nell'approccio base di una *k-fold CV* il training set è diviso in k insiemi più piccoli detti *fold*. Il modello è allenato su $k-1$ fold e valutato sul k-esimo per k volte cambiando ad ogni ciclo la fold di valutazione. Il modello risultante è validato sul test set. Questo approccio può essere costoso computazionalmente ma permette l'utilizzo di più dati in fase di allenamento.

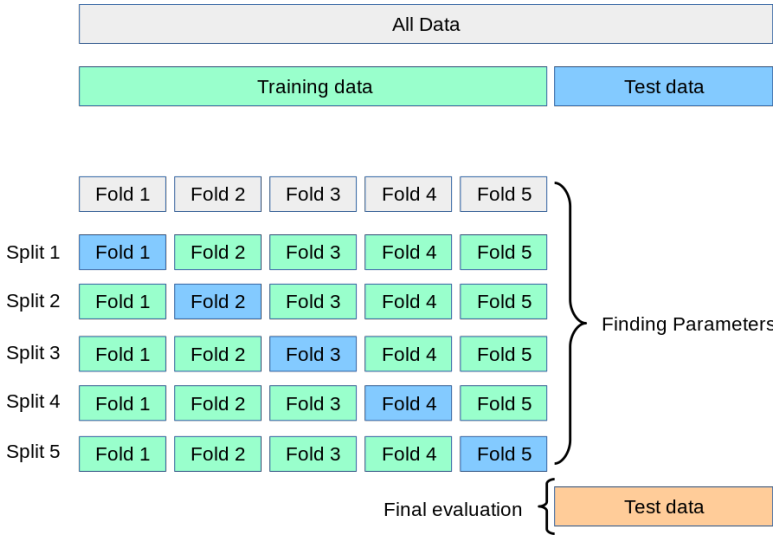


Figura 7.1: Esempio di Cross Validation con 5-folds

7.2.4 Test del modello

Questa fase è fondamentale per capire se il modello è adeguato per l'uso reale. Se le prestazioni del modello non sono soddisfacenti, è possibile provare a ottimizzare il modello cambiando i parametri di input o utilizzando un modello diverso, ripartendo, dunque dalla fase di selezione e addestramento.

Uno strumento utile in questa fase è chiamato *matrice di confusione*.

Matrice di confusione

Questa matrice è utile per visualizzare i risultati di test di un modello di classificazione. Essa riassume le previsioni del modello rispetto alle etichette reali ed è utile per capire in che modo il modello sta commettendo errori. Per un caso di etichette multiclasse la matrice avrà una dimensione $n \times n$ con $n = \#\text{classi}$. Ogni cella (i, j) rappresenta il numero di esempi della classe i che il modello ha classificato come classe j. I valori sulla diagonale sono quindi il numero di esempi classificati correttamente dal modello.

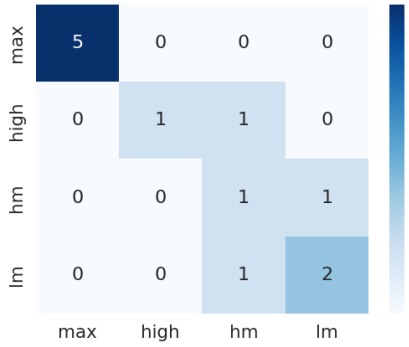


Figura 7.2: Esempio di Matrice di confusione

7.3 Algoritmi di apprendimento non supervisionato

Questo tipo di apprendimento non ha come scopo quello di predire il giusto output, bensì quello di esplorare e trarre inferenze dai set di dati. Gli algoritmi di apprendimento non supervisionato vengono usati quando le informazioni utilizzate per l'addestramento non sono etichettate, ovvero non è dato un output riconosciuto come obiettivo.

7.3.1 Principal Component Analysis (PCA)

L'analisi delle componenti principali è una tecnica di apprendimento non supervisionato anche usata per ridurre linearmente la dimensionalità dei dati. Tale riduzione è utile quando si vogliono eliminare dal dataset le

informazioni ridondanti, meno rilevanti per il problema da risolvere, o si vogliono semplicemente visualizzare i dati.

Nel caso di una matrice di dati $n \times p$ possiamo pensare a punti in uno spazio p-dimensionale. Se p è elevato l'obiettivo è quello di ridurre la rappresentazione dei punti ad uno spazio di dimensioni più ridotte ma che mantenga nel miglior modo possibile la struttura dei punti originari.
L'analisi delle componenti principali cerca dunque di individuare le combinazioni lineari delle variabili originali che catturino la maggior parte della varianza presente nei dati. Per fare questo utilizza le informazioni sulla covarianza delle variabili originali, considerando la matrice di covarianza dei dati. Questa matrice è tale per cui date le variabili X_1 e X_2 la cella corrispondente contiene il valore $cov(X_1, X_2)$. Qualora la matrice di covarianza si presenti diagonale o quasi diagonale, non si avrebbe correlazione tra i dati e dunque l'esecuzione di questa analisi non porterebbe nessun effetto in quanto la varianza totale da preservare sarebbe catturata considerando necessariamente tutte le variabili e non un loro sottoinsieme.
Nel caso in cui la matrice, invece, non sia diagonale, le componenti principali possono essere estratte calcolando gli autovettori. Questi delineano le direzioni in cui i dati variano, maggiore l'autovalore associato e maggiore la varianza trasportata: se le variabili originarie erano parzialmente correlate tra loro alcuni autovalori avranno un valore trascurabile ed è possibile dunque limitare la rappresentazione solo agli autovettori con gli autovalori più grandi [5].

Nel nostro caso l'esecuzione di una PCA verrà utilizzata durante la fase di pre-processing, per permetterci di provare un allenamento dei modelli su dati trasformati dall'applicazione dell'analisi delle componenti principali e verrà poi utilizzata per ridurre la dimensionalità di alcuni dati in modo da poterli visualizzare in due dimensioni.

7.3.2 Clustering e K-means

Il clustering è una tecnica algoritmica per l'apprendimento non supervisionato. L'idea è quella di raggruppare i dati in modo tale che dati simili siano contenuti all'interno dello stesso gruppo e che dati diversi appartengano a gruppi diversi. Esistono diversi algoritmi per raggiungere questo obiettivo, il più famoso prende il nome di K-means.
Poiché vogliamo gruppi di osservazioni tali che le osservazioni in un gruppo siano simili tra loro, un buon clustering è quello con la più bassa Sum of Squared Errors (SSE), cioè la più bassa somma della distanza al quadrato

da ogni osservazione al centroide del cluster a cui appartiene.

$$\text{SSE} = \sum_{i=1}^{N_C} \sum_{x \in C_i} d(x, \bar{x}_{C_i})^2 \tag{7.4}$$

dove:

- $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
  con $x = (x_1, x_2, ..., x_n)$ e $y = (y_1, y_2, ..., y_n)$ osservazioni di dimensione n
- N_C è il numero di cluster
- C_i è l' i-esimo cluster
- $\bar{x}_{C_i}$ è il centroide dell'i-esimo cluster

Come ottengo K? Si utilizza quello che viene chiamato *Elbow Criterion Method*. L'idea dietro a questo metodo è di eseguire il clustering con k-means sul dataset con valori di k in un dato range e per ogni valore di k calcolare SSE. Ad un più alto valore di k corrisponde necessariamente SSE più basso. Il metodo considerato dice che dovremmo scegliere k tale per cui aumentandolo non si ottiene una diminuizione più significativa del valore SSE, ad esempio in figura 7.3 la diminuizione dell'errore che si ottiene passando da $k = 3$ a $k = 4$ è inferiore della diminuzione che si ottiene passando da $k = 2$ a $k = 3$ e dunque in questo caso $k = 3$ risulterebbe essere il valore da utilizzare.

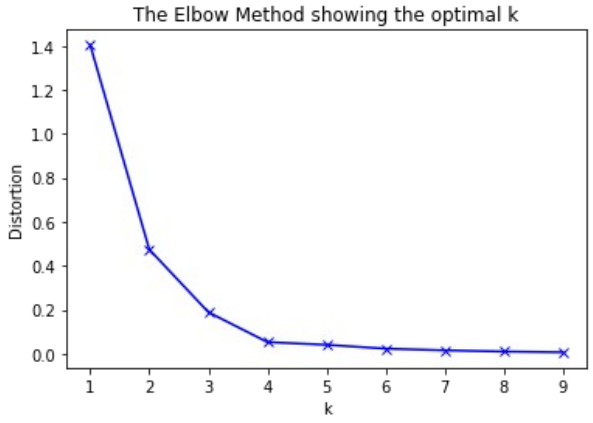


Figura 7.3: Esempio di applicazione del metodo del gomito

7.4 Algoritmi di apprendimento supervisionato

L'apprendimento supervisionato, al contrario di quello non supervisionato, è un tipo di apprendimento automatico in cui un modello è addestrato su un insieme di dati etichettati. Dopo l'allenamento, il modello impara a generalizzare e fare previsioni su dati mai visti in precedenza. Vengono presentati adesso due algoritmi utilizzati durante questo studio.

7.4.1 K-nearest neighbors

L'algoritmo KNN assume che dati simili siano vicini tra loro. L'addestramento di un modello che utilizza KNN avviene su dati etichettati e quando un nuovo esempio viene presentato, vengono cercate le k osservazioni più simili (i k nearest neighbors) utilizzando una misura di similarità come, ad esempio, la distanza Euclidea:

$$d(P, Q) = \sqrt{\sum_{i=1}^{n} (P_i - Q_i)^2}$$

Si osserva che l'utilizzo di questo algoritmo può portare ad overfitting sui dati presenti in maggioranza. In questo caso può risultare utile pesare i contributi dei vicini in modo da dare, nel calcolo della media, maggior importanza in base alla distanza dall'oggetto considerato.

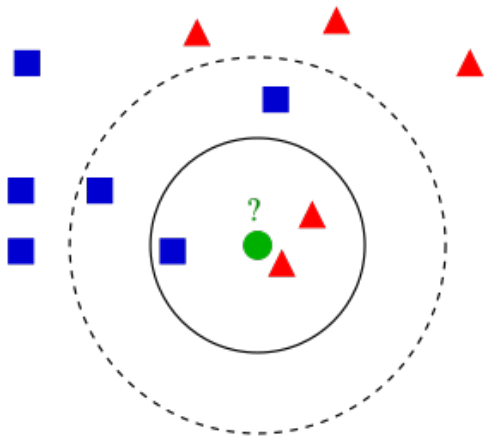


Figura 7.4: Esempio di selezione della classificazione di un sample con KNN

7.4.2 Random Forest

L'algoritmo Random Forest, come il nome suggerisce, crea un insieme di Decision Tree, dove ogni albero viene costruito usando un sottoinsieme casuale di caratteristiche del set di addestramento.
Un **Decision Tree** è un tipo di modello utilizzato sia per la classificazione che per la regressione. Gli alberi rispondono a domande sequenziali che, data la risposta, indirizzano verso un determinato percorso rispetto ad un altro. La profondità dell'albero è un concetto importante: rappresenta il numero di domande che vengono poste prima di raggiungere la classificazione prevista. Quando l'albero è particolarmente profondo l'albero è incline ad overfitting perché ha una grande quantità di nodi, rischiando di essere troppo sensibile alle variazioni nei dati di allenamento, a discapito della capacità di generalizzare a dati sconosciuti. Nell'utilizzo della tecnica Random Forest ogni albero di decisione viene addestrato su un sottoinsieme diverso dei dati di allenamento, rendendo meno probabile che la foresta soffra di overfitting rispetto ad un singolo albero e producendo quindi risultati utili.

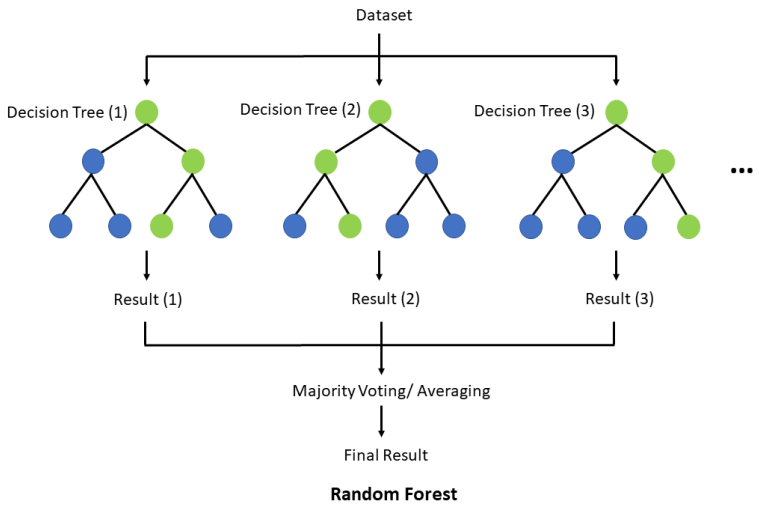


Figura 7.5: Esempio di Random Forest con tre DT

8. Preprocessing dei dati

In questo capitolo verrà descritto il lavoro svolto ed i risultati ottenuti in fase di preprocessing dei dati. Da non sottovalutare, la qualità del primo approccio ai dati determina l'affidabilità della fase di analisi. Ottenere i dati giusti è il primo passo di un qualsiasi progetto di Machine Learning e spesso richiede più tempo e complessità del perfezionamento degli algoritmi di apprendimento stessi.

8.1 Comprensione dei dati

Siamo provvisti di dati ottenuti dall'aggregazione dei dati grezzi collezionati dagli attigrafi utilizzati. Questi dispositivi registrano i dati accelerometrici ad una frequenza preimpostata e permettono l'esportazione e la conversione di essi, tramite software ActiLife, in Activity Counts, ovvero una misura del movimento in un certo asse. Ad un valore più alto corrisponde un movimento con maggiore intensità, o più movimenti.
Ai soggetti è stato chiesto di indossare gli attigrafi ai polsi durante due sessioni diverse: una in laboratorio, della durata di 20 minuti circa, ed una a casa, della durata di una settimana circa.

8.1.1 Dati di laboratorio

Durante le sessioni di laboratorio, ai soggetti è stato chiesto di (o sono stati portati a) eseguire dei movimenti predefiniti, indossando attigrafi impostati ad una frequenza di 80Hz. Su questi dati si è concentrato l'allenamento dei modelli. Nel seguito per riferirsi a questo tipo di sessioni sarà utilizzato il termine "sessioni AHA".

8.1.2 Dati a casa

I dati a casa sono stati ottenuti da due attigrafi indossati dai soggetti durante la giornata, nell'arco di una settimana. La frequenza impostata e il numero di Activity Counts ottenuti è lo stesso che per le sessioni di laboratorio. Nel seguito, per riferirsi a questo tipo di sessioni sarà utilizzato il termine "sessioni WEEK".

8.1.3 Aggregazione dei dati

I dati a disposizione sono stati divisi in finestre: ognuna di esse è un'osservazione nel contesto dell' allenamento del modello. Le caratteristiche di ogni finestra sono le seguenti:

- La media aritmetica dei dati
- La deviazione standard, ovvero la misura della dispersione dei dati intorno alla media aritmetica
- Il 25-esimo, 50-esimo e il 75-esimo percentile, ovvero, considerato un percentile q-esimo, il minimo valore sotto al quale ricade il q% degli altri elementi sotto osservazione
- L' asimmetria (skewness), ovvero una misura dell'asimmetria della distribuzione di probabilità di una variabile a valori reali rispetto alla sua media

8.1.4 Bilanciamento e dimensione del dataset

Si vuole allenare i modelli con i dati relativi alle sessioni AHA e predire la valutazione sui dati relativi alle sessioni WEEK. Ogni finestra estratta è stata etichettata con una classe artificiale derivata dalla valutazione AHA relativa al soggetto considerato. È stato convenuto, in accordo anche con i ricercatori della fondazione Stella Maris, che per un miglior bilanciamento del dataset fosse necessaria la divisione in classi della valutazione iniziale, originariamente un valore continuo. Le etichette risultanti dopo questo procedimento sono riportate nella tabella 8.1 insieme al numero di soggetti per classe.
Il dataset a disposizione si compone di poche osservazioni, che sono le finestre estratte dalle sessioni AHA dei soggetti. Questo comporta la scelta, in fase di analisi, di allenare modelli classici di machine learning invece di modelli che sfruttano il deep learning: in generale, gli algoritmi di deep learning

Classe	Intervallo AHA	# soggetti
Low moderate	$[39, 49]$	12
High moderate	$[50, 62]$	10
High	$[63, 99]$	11
Max	$\{100\}$	26

Tabella 8.1: Bilanciamento del dataset

funzionano in modo più accurato su grandi set di dati rispetto agli algoritmi di machine learning, mentre questi ultimi superano il deep learning in caso di insiemi di dati di piccole o medie dimensioni [8].

8.2 PCA

I dati a disposizione sono stati esplorati per esaminare alcuni aspetti, tra i quali, la distribuzione dei dati contenuta in ciascuna variabile.

La figura 8.1 è il risultato dell'esecuzione di una PCA sulle finestre delle sessioni AHA relative a 60 soggetti; questa quantifica quanta parte della varianza totale è contenuta nelle prime n componenti. È possibile notare che sono necessarie almeno 3 componenti per preservare più del 95% della varianza: questo ci indica che vale la pena provare l'allenamento dei modelli sui dati trasformati dall'analisi.

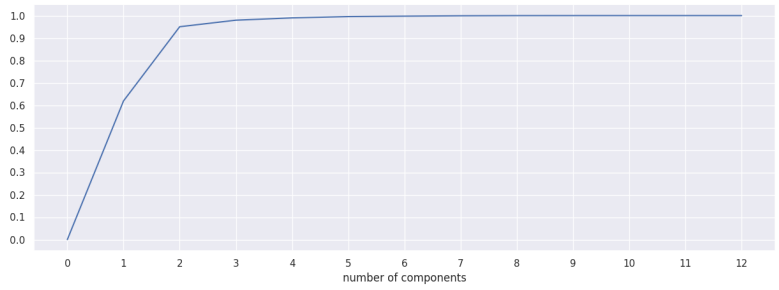


Figura 8.1: Percentuale della varianza in funzione del numero di componenti

Il risultato trovato è in linea con la matrice di covarianza che si ottiene dai dati (figura 8.2), infatti non si ha una matrice diagonale, sintomo che c'è correlazione tra i dati a nostra disposizione. La covarianza infatti, fornisce

una misura della relazione lineare tra le variabili e, se diversa da 0, indica una correlazione tra di esse.

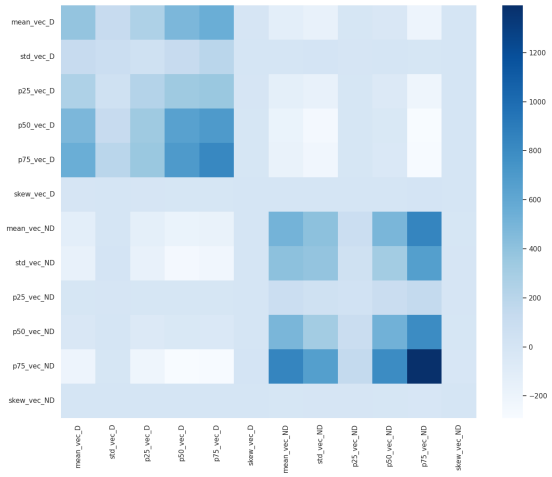


Figura 8.2: Matrice di covarianza dei dati originali

8.3 Scaling dei dati

Lo scaling dei dati prevede il riscalamento degli stessi attraverso un calcolo conosciuto come *standardizzazione*.

La standardizzazione ridimensiona il valore delle features in modo da avere una distribuzione con valore medio pari a 0 e varianza pari a 1.

$$X_{\text{new}} = \frac{X_i - X_{\text{mean}}}{\sigma} \tag{8.1}$$

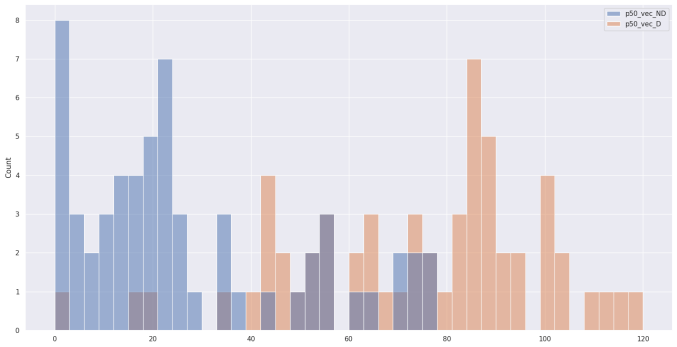
dove:

X_i è il valore i-esimo della feature
X_{mean} è il valore medio della feature
σ è la deviazione standard

Prima di procedere a questo tipo di scaling dei dati è necessario considerare che esistono delle situazioni in cui questo processo non porta benefici ma addirittura può presentare degli svantaggi. Si consideri, ad esempio, di

avere un dataset con delle features con distribuzioni molto diverse tra loro. La standardizzazione di queste porterebbe le feature con distribuzione più ampia ad avere una varianza molto più piccola di quella reale rispetto alle feature con distribuzione più stretta, le cui varianze verrebbero ridotte di meno. Tutto ciò può rendere il modello meno sensibile alle prime.

Nel nostro caso si può notare, ad esempio, la differenza di distribuzione tra il 50-esimo percentile relativo alla magnitudo del vettore del braccio dominante e quello relativo al braccio non dominante: la distribuzione del primo varia nel range $[0, 120]$, molto più ampio rispetto al secondo $[0, 78]$.



9. Analisi

In questo capitolo verrà descritto il lavoro svolto ed i risultati ottenuti in fase di analisi dei dati. L'obiettivo iniziale è quello di allenare un modello nell'ottica di affiancarlo ai medici per rendere disponibili loro informazioni ulteriori circa la condizione dei bambini. In secondo luogo si cerca di classificare le attività che vengono svolte dai soggetti a casa.

9.1 Allenamento di classificatori

Sono state sperimentate diverse soluzioni. Due di queste sono apparse migliori. In linea generale, l'addestramento con dati standardizzati ha portato ad esiti peggiori, mentre quello con dati trasformati tramite PCA, preservando almeno il 95% della varianza totale, ha dato risultati più soddisfacenti. I due algoritmi con punteggio maggiore sono stati *KNN* e *Random Forest*.

9.1.1 K Nearest Neighbors

L'implementazione del classificatore KNN utilizzata permette l'ottimizzazione attraverso la sintonizzazione di alcuni parametri:

- n_neighbors: il valore K. E' il numero di vicini meno distanti da interrogare.
- metric: metrica da utilizzare per il calcolo della distanza.
- weights: se sia meglio ponderare i vicini, in modo che quelli meno distanti contribuiscano maggiormente all'adattamento, è una scelta effettuata attraverso il valore di questo parametro.

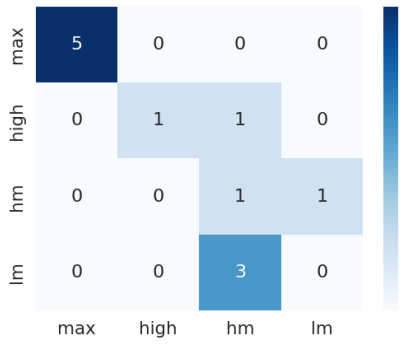
La tabella 9.1 si riferisce ai risultati di test ottenuti dal miglior modello estrapolato da una funzione di ricerca che utilizza il Cross-Validation e confronta modelli con iperparametri diversi. I parametri trovati dalla funzione sono i seguenti:

- n_neighbors: 11.
- metric: manhattan.
- weights: distance, ovvero i pesi assegnati sono proporzionali all'inverso della distanza dal punto di interrogazione.

Classe	Precisione	Recall	F1 score	support
max	1.00	1.00	1.00	5
high	1.00	0.50	0.67	2
high moderate	0.20	0.50	0.29	2
low moderate	0.00	0.00	0.00	3
accuracy			0.58	12

Tabella 9.1: Classificatore KNeighbors

Si nota un possibile overfitting sui dati della classe max. I dati sono visualizzati nella seguente matrice di confusione:



9.1.2 Random Forest

L'implementazione del classificatore Random Forest utilizzata permette l'ottimizzazione attraverso la sintonizzazione di alcuni parametri:

- n_estimators: il numero di alberi nella foresta.
- max_features: il numero massimo di caratteristiche che possono essere considerate per la separazione di un nodo.

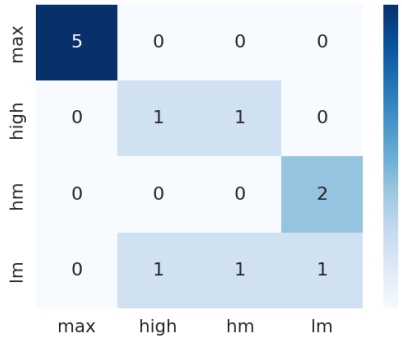
Anche in questo caso, la tabella 9.2 si riferisce ai risultati di test ottenuti dal miglior modello estrapolato da una funzione di ricerca che utilizza il Cross-Validation e confronta modelli con iperparametri diversi. I parametri trovati dalla funzione sono i seguenti:

- n_neighbors: 1000.
- max_features: sqrt, ovvero il numero di feature dei DT va con la radice quadrata del numero totale di feature.

Classe	Precisione	Recall	F1 score	support
max	1.00	1.00	1.00	5
high	0.50	0.50	0.50	2
high moderate	0.00	0.00	0.00	2
low moderate	0.33	0.33	0.33	3
accuracy			0.58	12

Tabella 9.2: Classificatore Random Forest

I dati sono visualizzati nella seguente matrice di confusione:



9.2 Clustering

La ricerca di informazioni relative allo svolgimento delle attività giornaliere dei soggetti è iniziata utilizzando l'algoritmo K-means per effettuare clustering sui dati delle sessioni week, suddivisi in finestre da 15 minuti. In questo modo sono state cercate delle relazioni tra i dati. Successivamente è stato effettuato un clustering manuale, raggruppando le finestre delle sessioni week di ogni soggetto in insiemi (cluster) tali per cui ogni insieme contenga i dati registrati nello stesso periodo della giornata. I periodi a cui si fa riferimento sono intervalli di tempo di 2 ore: $[0, 2), [2, 4), \ldots, [22, 24)$.

9.2.1 Classificatori su cluster

I modelli mostrati nella sezione precedente sono stati utilizzati per predire le osservazioni all'interno di ogni cluster. Questo ci ha permesso di studiare la capacità di predizione all'interno di gruppi di dati simili tra loro.

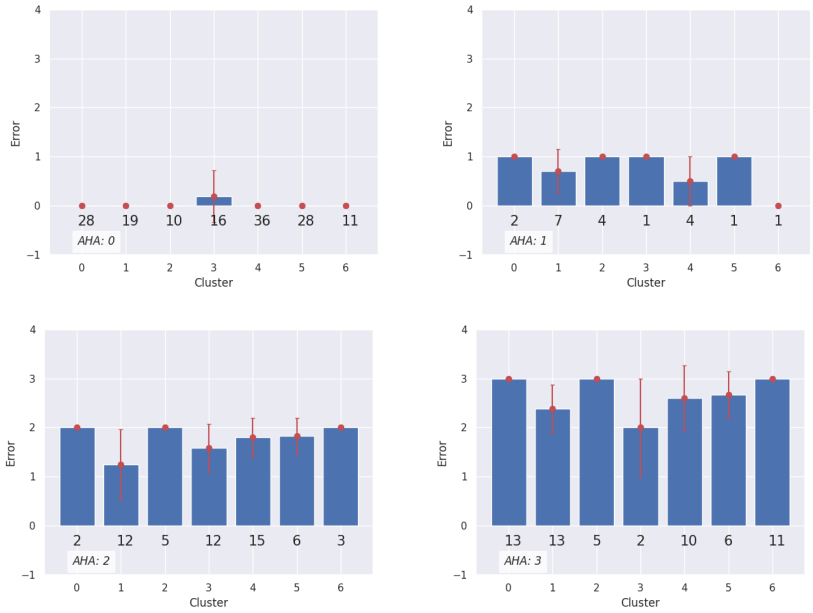
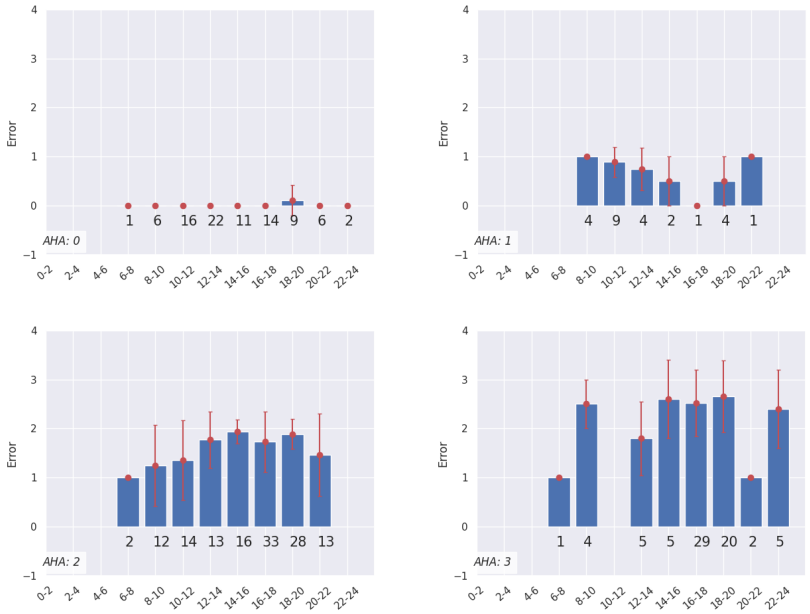


Figura 9.1: Gli errori medi, con deviazione standard, delle quattro classi AHA. Per ogni cluster è riportata la dimensione.

Per questo studio sono stati rappresentati graficamente gli errori medi su

ogni cluster per ogni soggetto. I grafici visualizzati sono solo alcuni elementi dell'insieme di rappresentazioni prodotte per questo tipo di lavoro.
Come è possibile notare, purtroppo i modelli presentano una tendenza a classificare come classe 0 (soggetti sani) le finestre delle sessioni week date loro come input. I risultati delle predizioni mostrano infatti che l'errore aumenta proporzionalmente alla classe AHA reale dei bambini e alla dimensione del cluster considerato.

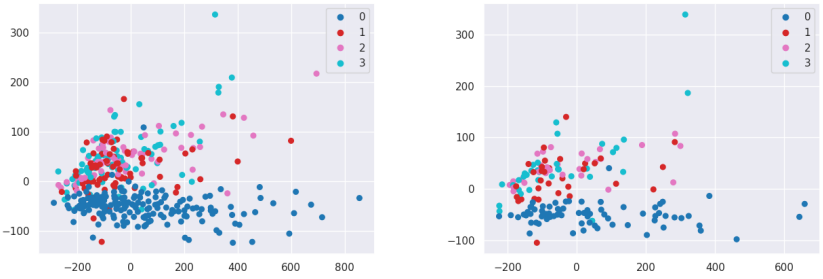
Lo stesso tipo di conclusioni possono essere tratte dalle predizioni effettuate sui cluster risultanti dal raggruppamento in intervalli orari:



Purtroppo, non è stato possibile ricavare dettagli importanti dai cluster costruiti manualmente: osservando i risultati non si ha alcuna evidenza, infatti, che ci siano cluster (o intervalli orari) migliori di altri in termini di informazioni utili a questo studio.

9.2.2 Rappresentazione grafica dei cluster

Abbiamo utilizzato **k-means** in due modi: con k fissato e con k ricavato con elbow-method per ogni soggetto. Per entrambi i metodi abbiamo riportato su un grafico in due dimensioni i centroidi di ogni cluster colorando in modo uguale i centroidi appartenenti a cluster relativi a soggetti della stessa classe AHA. I grafici mostrano la capacità del metodo di riconoscere i sani



dagli altri in quanto è ben visibile una divisione netta dello spazio tra questi due tipi di soggetti. Si nota infatti che i centroidi appartenenti alla classe 0 (classe con valutazione AHA massima, quella dei soggetti sani) condividono una parte definita dello spazio, diversa da quella in cui sono collocati tutti gli altri.
Nel caso del **clustering per intervalli di tempo** i centroidi sono stati calcolati manualmente, ovvero è stato calcolato il punto medio tra tutti i punti all'interno di un cluster fissato, e si è tracciato un grafico analogo a quelli mostrati precedentemente. In questo caso si è anche provato a dividere le varie attività dei soggetti (rappresentate dai centroidi trovati) non secondo le classi finora considerate, bensì in base ai vari periodi della giornata, attribuendo una colorazione diversa per ogni fascia oraria. L'ultima

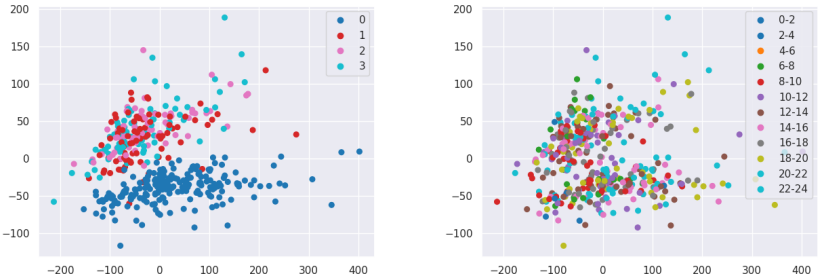


figura mostra come la divisione per classe AHA suddivida lo spazio in modo analogo al caso precedente e come, invece, la divisione per fascia oraria non evidenzi alcuna somiglianza tra le attività registrate.

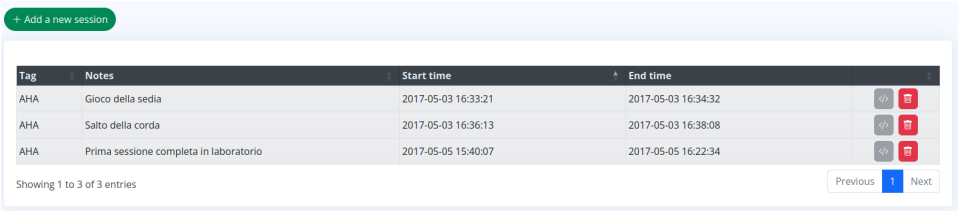
Parte IV

Conclusioni

10. Sviluppi futuri

Per quanto riguarda la **piattaforma web**, eventuali aggiornamenti consisterebbero nella costruzione di un nuovo prototipo. È stato già evidenziato un problema prestazionale nel capitolo conclusivo ma altre limitazioni sono presenti nella prima versione:

- l'aggiunta di una sessione viene sempre preceduta dall'upload di un file perchè non è ancora presente la possibilità di scegliere tra i file già caricati precedentemente.
- il sistema è esposto ad attacchi CSRF per quanto riguarda l'eliminazione di soggetti e sessioni in quanto queste operazioni sono lanciate in risposta a richieste GET.
- una volta salvata una sessione di un bambino attraverso il form apposito non è ancora possibile visualizzarla nuovamente al di fuori della tabella che contiene le sessioni del soggetto.



- l'interfaccia grafica può essere migliorata per risultare più chiara fin dal primo accesso. Si potrebbe approfondire lo studio della UX per passare poi alla creazione di un wireframe da seguire in fase di implementazione dell'interfaccia.

In termini di funzionalità, ai ricercatori risulta utile accedere al valore dell'indice di asimmetria (AI) direttamente dalla dashboard. Si può dunque considerare di implementare il calcolo dell'indice per ogni sessione catalogata nel sistema.

Quelli presentati potrebbero essere dei buoni punti di partenza per lo sviluppo di un secondo prototipo. Il codice sorgente della piattaforma è consultabile su github [9].

Per quanto riguarda il lavoro di **analisi dei dati**, purtroppo è stata riscontrata l'inadeguatezza dei classificatori alla predizione delle classi AHA. Un motivo potrebbe essere l'esiguo numero di dati utilizzabili per l'allenamento o la loro qualità. L'ausilio della piattaforma sviluppata potrebbe essere utile a migliorare la precisione dei dati: si potrebbe infatti etichettare le singole attività all'interno di una sessione AHA ed esportare le specifiche finestre da utilizzare in fase di addestramento del modello. I dati a casa potrebbero essere suddivisi in intervalli più brevi per assecondare la riduzione della dimensione delle finestre di allenamento e si potrebbe, a quel punto, avere una predizione migliore su alcuni cluster, che trasporterebbero informazioni più precise. Sarebbe dunque possibile concentrare il peso delle predizioni sui cluster più informativi.

11. Commenti personali

In questo studio abbiamo implementato una pipeline di lavoro a supporto dei medici e dei ricercatori inerentemente al progetto AInCP. È stato necessario lavorare dapprima in un contesto rigido e freddo quale lo sviluppo di una piattaforma web, per poi navigare nei mari della ricerca e dello studio di algoritmi di machine learning (sperabilmente) utili alla causa. La navigazione tra queste onde non è stata lineare e ci ha reso inizialmente insicuri di voler continuare la traversata, ma una volta consci delle correnti siamo diventati man mano consapevoli che non c'è mare che una barca non possa solcare. Tuttavia, le modeste conoscenze del sottoscritto hanno permesso di esplorare solo poche delle infinite rotte possibili.
Si capisce che ci vuole ben altro: (ma) siamo felici [1].

[1] Alla memoria di Carlo Casolo, professore del dipartimento di Matematica di Firenze, che in poco tempo ha saputo trasmettere conoscenze e passione.

Bibliografia

[1] Ankit Mehta (Google). *Move Fast & Don't Break Things*. URL: https://docs.google.com/presentation/d/15gNk21rjer3xo-b1ZqyQVGebOp_aPvHU3YH7YnOMxtE/edit#slide=id.g437663ce1_53_98.

[2] Beani E. et al. "Actigraph assessment for measuring upper limb activity in unilateral cerebral palsy." In: *Journal of neuroengineering and rehabilitation* 16 (2019).

[3] Krumlinde-Sundholm L et al. "The assisting hand assessment: current evidence of validity, reliability, and responsiveness to change." In: *Dev Med Child Neurol* 49(4) (2007), pp. 59–64.

[4] Martin Bax et al. "Proposed definition and classification of cerebral palsy". In: *Dev Med Child Neurol* (2005).

[5] E. Martinell & iC. Di Natale. *Introduzione al trattamento Statistico dei Dati*. URL: https://didattica-2000.archived.uniroma2.it//Sensori_ed_applic_2/deposito/lez5.pdf.

[6] Laura Semini Giovanni A. Cignoni Carlo Montangero. *Il controllo software: verifica e validazione*. URL: http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/is-a/dispensatestingconv.pdf.

[7] *GT3X — ActiGraph*. URL: https://www.theactigraph.com/support/activity-monitors/gt3x/.

[8] Ayushi Chahal & Preeti Gulia. *Machine Learning and Deep Learning*. URL: https://www.researchgate.net/profile/Preeti-Gulia-3/publication/364097061_Machine_Learning_and_Deep_Learning/links/633c0fceff870c55cefe1adc/Machine-Learning-and-Deep-Learning.pdf.

[9] *Piattaforma web AInCP su Github*. URL: https://github.com/mikeabbott10/AIncP-platform.

[10] *Progetto europeo AINCP*. URL: https://www.fsm.unipi.it/2022/03/21/progetto-europeo-aincp-la-coordinatrice-scientifica-dr-ssa-giuseppina-sgandurra-ci-spiega-gli-obiettivi/.

[11] *Verifica e Validazione del Software - Testing Black Box*. URL: https://www.docenti.unina.it/webdocenti-be/allegati/materiale-didattico/639711.

[12] Mike Wacker. *Just Say No to More End-to-End Tests*. URL: https://testing.googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html.