UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Tesi di Laurea in Informatica

Tecnologia e salute mentale: un sistema per il monitoraggio e l'assistenza ai pazienti schizofrenici

Relatore:
Prof.ssa
Rita Francese

Candidato:
Gianpio Silvestri
Mat. 0512107818

ANNO ACCADEMICO 2023/2024

SOMMARIO

La schizofrenia è un disturbo mentale complesso che influisce profondamente sulla capacità di pensare e agire in modo lucido, con conseguenze significative sulla qualità della vita dei pazienti. I sintomi, tra cui pensiero disorganizzato, isolamento sociale, deliri e allucinazioni, non sono statici e possono variare nel tempo, portando a periodi di peggioramento o remissione. Attualmente, i pazienti ricevono supporto da centri specializzati, ma l'assistenza post-dimissione è limitata, esponendoli a un elevato rischio di ricadute. Il sistema proposto si prefigge di monitorare continuamente i pazienti schizofrenici tramite smartphone, sfruttando i sensori integrati per raccogliere dati relativi alle loro attività quotidiane. Questi dati vengono inviati a un server dedicato, dove vengono elaborati e archiviati in un database. Utilizzando un modello predittivo basato su autoencoder e Long Short-Term Memory (LSTM), il sistema è in grado di rilevare anomalie comportamentali e prevedere potenziali ricadute. Si prevedono sviluppi futuri che includono l'integrazione di dati provenienti da dispositivi indossabili per aumentare la precisione del monitoraggio e l'implementazione di sondaggi regolari, che consentirebbero una valutazione più dettagliata e diretta dello stato di salute mentale dei pazienti. Questo approccio mira a fornire un intervento proattivo nella gestione della malattia, contribuendo così a migliorare il benessere dei pazienti e a supportare i professionisti nella gestione della loro assistenza.

INDICE

CAPITOLO 1

INTRODUZIONE

La schizofrenia (SZ) è una malattia caratterizzata da alterazioni della percezione, pensiero disorganizzato, isolamento sociale, deliri, allucinazioni e difficoltà cognitive come memoria e attenzione. La SZ colpisce circa 24 milioni di persone o 1 persona su 300 (0,32%) in tutto il mondo. Questo tasso diventa di 1 persona su 222 (0,45%) se consideriamo solo la popolazione adulta. Nella maggior parte dei casi i primi sintomi tendono ad insorgere tra la tarda adolescenza ed i vent'anni, in maniera più precoce negli uomini rispetto alle donne [1]. Esiste una serie di opzioni di cura efficaci per le persone affette da SZ ed almeno una persona su tre sarà in grado di guarire completamente [2]. Nonostante ciò, più di due persone su tre che soffrono di SZ non ricevono le cure specialistiche necessarie per il trattamento della malattia. Un'adeguata gestione terapeutica della SZ è necessaria per il trattamento della malattia. Tuttavia, nonostante un trattamento attivo, una persona potrebbe sperimentare un peggioramento continuo dei sintomi nel tempo, oppure un peggioramento alternato ad una remissione dei sintomi in maniera periodica [1]. I clinici devono monitorare lo stato dei sintomi dei pazienti affetti da SZ per identificare i rischi e adattare il trattamento, se necessario. L'intervallo tra le visite varia da una volta a settimana a una volta al mese, a seconda della gravità

dei sintomi e del rischio dei pazienti. Durante le visite i sintomi dei pazienti vengono monitorati tramite dei questionari, come la Brief Psychiatric Rating Scale (BPRS) [3]. Tuttavia, a causa della frequenza delle visite, un paziente potrebbe sperimentare un peggioramento o una ricomparsa dei sintomi tra una visita e l'altra. Questo crea un notevole peso sulla vita quotidiana del paziente e potrebbe addirittura rappresentare un pericolo per la salute sua e di chi gli sta intorno. Pertanto, un metodo automatizzato per la valutazione dei sintomi, basato sul comportamento del paziente, potrebbe essere utile per prevedere una ricaduta imminente, poiché i cambiamenti comportamentali sono associati a ricadute in arrivo [4]. Lo smartphone si rivela uno strumento ideale per la previsione delle ricadute, grazie alla sua ampia diffusione e al suo utilizzo quotidiano. Inoltre, sfruttando i sensori di uno smartphone come GPS, accelerometro, livello di luce ambientale ecc. è possibile modellare i comportamenti di un paziente, in modo implicito o esplicito [5]. Sfruttando i dati ricavati dai sensori di uno smartphone, è possibile addestrare un modello per la previsione delle ricadute di un paziente, che potrebbe informare medici e/o familiari in modo da intervenire in maniera proattiva e prevenire esiti indesiderati.

1.1 Struttura del documento

Il seguente documento di tesi, che riassume il lavoro svolto, è organizzato come segue:

- **Capitolo 2 - Background**: In questo capitolo viene fornito un contesto teorico per comprendere i temi trattati nel lavoro di tesi. Nello specifico, viene fatta una panoramica sulla scala BPRS, sulla necessità di monitoraggio continuo dei pazienti affetti da schizofrenia e sul digital phenotyping. Viene inoltre esaminato lo stato dell'arte sull'uso degli smartphone per la raccolta di dati utili nella previsione delle ricadute nei pazienti schizofrenici.

- **Capitolo 3 - Tecnologie utilizzate**: In questo capitolo verrà fatta una panoramica sulle tecnologie utilizzate per lo sviluppo del sistema proposto.
- **Capitolo 4 - Soluzione proposta**: In questo capitolo verrà presentata la soluzione proposta per il monitoraggio remoto dei pazienti schizofrenici composta da un'applicazione mobile, un server per l'elaborazione dati ed un modello per la predizione delle ricadute.
- **Capitolo 5 - Modello di predizione**: In questo capitolo verrà illustato dettagliatamente il modello di predizione implementato, con un focus sulle diverse componenti e le decisioni progettuali intraprese.
- **Capitolo 6 - Implementazione**: In questo capitolo verrà dettagliata l'implementazione delle varie componenti del sistema.
- **Capitolo 7 - Conclusioni e sviluppi futuri**: In questo capitolo verranno sintetizzate le conclusioni del lavoro svolto, evidenziando le implicazioni dei risultati ottenuti e suggerendo possibili sviluppi futuri per il sistema.

CAPITOLO 2

BACKGROUND

In questo capitolo verranno affrontati i temi centrali del lavoro di ricerca: la scala BPRS, utilizzata per valutare la gravità dei sintomi nei pazienti affetti da schizofrenia; l'importanza del monitoraggio continuo per garantire un'assistenza efficace e tempestiva a questi pazienti e il concetto di digital phenotyping, che permette di analizzare il comportamento attraverso i dati raccolti dai dispositivi digitali. A conclusione, verrà fornita una panoramica aggiornata sullo stato dell'arte riguardante l'uso di smartphone per la previsione delle ricadute nei pazienti schizofrenici.

2.1 La scala BPRS

La Brief Psychiatric Rating Scale (BPRS) è uno strumento clinico ampiamente utilizzato per valutare la gravità dei sintomi psicopatologici in pazienti con disturbi mentali, in particolare la schizofrenia, i disturbi schizoaffettivi e altre gravi patologie psichiatriche. Sviluppata da Overall e Gorham nel 1962, la scala include 18 item che misurano una vasta gamma di sintomi tra cui sintomi positivi (come allucinazioni e deliri), sintomi negativi (come apatia e ritiro sociale), nonché manifestazioni affettive (ansia e depressione)

e comportamentali (ostilità, sospettosità). Ogni item è valutato su una scala a 7 punti, dove 1 indica l'assenza del sintomo e 7 rappresenta la massima gravità. La somministrazione della BPRS richiede una visita clinica o un'intervista strutturata con il paziente, durante la quale il clinico osserva e valuta i sintomi attraverso domande specifiche e il comportamento del paziente. Questo metodo permette di ottenere una valutazione più accurata e completa della sintomatologia, garantendo che vengano rilevati sia i sintomi riferiti dal paziente sia quelli osservati dal clinico. La BPRS è ampiamente impiegata sia in contesti clinici che di ricerca per monitorare l'andamento del disturbo e l'efficacia del trattamento, fornendo un indicatore quantitativo e standardizzato. Grazie alla sua capacità di coprire un ampio spettro di sintomi la BPRS è particolarmente utile per studi longitudinali e per confrontare l'efficacia di diversi interventi terapeutici. Inoltre, la scala facilita una diagnosi più precisa e una gestione clinica mirata, migliorando così l'assistenza al paziente.

2.2 Monitoraggio continuo dei pazienti schizofrenici

Come già anticipato nell'introduzione, la schizofrenia è una patologia psichiatrica complessa che può manifestarsi con un peggioramento graduale e costante dei sintomi o attraverso cicli di peggioramento e remissione, anche quando il paziente è sottoposto a un trattamento continuo. La scala BPRS, introdotta nella sezione precedente, è uno strumento di valutazione estremamente utile per misurare i sintomi e comprendere lo stato clinico di un paziente. Tuttavia, la sua applicazione richiede la presenza di un medico specialista, il che implica che la valutazione del paziente avvenga durante visite programmate, solitamente a cadenza mensile. Questo intervallo temporale può rappresentare un problema significativo: il paziente potrebbe attraversare una fase di remissione o, peggio, un peggioramento rapido e non trattato dei sintomi, con il rischio di degenerare in condizioni più gravi. L'esigenza

di monitorare un paziente schizofrenico in modo più costante e tempestivo nasce proprio da questa difficoltà. Un approccio tradizionale al problema è quello di inserire il paziente in un programma di trattamento presso strutture specializzate per la cura della schizofrenia. In queste strutture il paziente riceve un supporto clinico e psichiatrico continuo, che consente un controllo regolare dei sintomi. Tuttavia, non è realistico mantenere un paziente all'interno di tali strutture per lunghi periodi, poiché l'obiettivo è spesso quello di restituire al paziente una certa autonomia nella vita quotidiana. Quando il paziente viene dimesso o ritorna a vivere nella sua comunità, il rischio di una ricaduta o di un peggioramento non diminuisce. In assenza di un monitoraggio costante, i sintomi possono evolvere senza che venga identificato tempestivamente un intervento adeguato. Questo circolo vizioso di miglioramenti seguiti da peggioramenti rappresenta una sfida critica nel trattamento a lungo termine della schizofrenia e sottolinea la necessità di sviluppare soluzioni innovative per il monitoraggio remoto dei pazienti. Un sistema che consenta di seguire l'andamento clinico del paziente al di fuori delle strutture sanitarie potrebbe ridurre significativamente il rischio di ricadute intervenendo in maniera precoce e prevenendo un aggravamento del quadro clinico.

2.3 Digital phenotyping

Il digital phenotyping è un campo scientifico multidisciplinare che si occupa della raccolta e analisi di dati comportamentali tramite dispositivi digitali come smartphone e wearable. Questo approccio, introdotto nel 2016 da John Torous e collaboratori, è stato definito come *"la quantificazione momento per momento del fenotipo umano attraverso dati raccolti in tempo reale da dispositivi personali"* [6]. I dati raccolti nel contesto del digital phenotyping possono essere suddivisi in due categorie principali: dati attivi e dati passivi. I dati attivi richiedono la partecipazione diretta dell'utente, come la risposta a questionari o il completamento di attività. I dati passivi, invece, vengono acquisiti senza intervento consapevole da parte

dell'utente, sfruttando i sensori interni dei dispositivi digitali. Tra i vari dispositivi utilizzati, lo smarthphone si presta particolarmente bene al digital phenotyping data la sua diffusa adozione e proprietà nel mondo moderno, oltre che alla presenza di sensori quali GPS, accelerometro, sensore di luce ecc. che permettono la raccolta di dati senza che l'utente debba necessariamente partecipare. L'adozione del digital phenotyping richiede linee guida metodologiche solide. La raccolta passiva dei dati presenta sfide in tutto il processo di ricerca, dalla definizione chiara dei costrutti da indagare, data la complessità dei fenomeni digitali, all'utilizzo dei dispositivi di acquisizione dei dati, delle applicazioni e dei protocolli di pulizia. Nella fase di analisi l'utilizzo di tecniche computazionalmente impegnative come l'apprendimento automatico introduce nuove sfide. L'ottimizzazione delle prestazioni del modello richiede una suddivisione accurata dei dati e la regolazione degli iperparametri, che richiedono una conoscenza approfondita. Recentemente sono stati proposti modelli che forniscano approcci standardizzati per la ricerca sulla fenotipizzazione digitale, al fine di favorire una maggiore coerenza e comparabilità tra gli studi [7].

2.4 Stato dell'arte

Negli ultimi anni, diverse piattaforme di digital phenotyping sono state sviluppate per la ricerca clinica e accademica. Tra le prime spicca il Funf Open Sensing Framework, lanciato dal MIT Media Lab nel 2011 [8], seguito da strumenti come Purple Robot e Beiwe [9], con quest'ultima sviluppata presso la Harvard School of Public Health nel 2013. Queste piattaforme raccolgono dati da dispositivi mobili, permettendo di analizzare i comportamenti individuali in tempo reale. Il primo tentativo significativo di utilizzare il digital phenotyping per monitorare la salute mentale risale al 2012 con il progetto BeWell [10]. L'obiettivo era migliorare il benessere generale degli individui e identificare i primi segni di declino. Per quanto riguarda l'applicazione di queste tecniche al trattamento di pazienti schizofrenici, il primo vero

tentativo di creare un sistema in grado di monitorare costantemente i pazienti è CrossCheck [5], uno studio in cui 37 pazienti ambulatoriali recentemente dimessi venivano monitorati per periodi di 2-12 mesi. Utilizzando i sensori degli smartphone, il sistema raccoglieva metriche comportamentali, come mobilità e conversazioni, integrandole con sondaggi inviati ai pazienti per prevedere le ricadute. Per la predizione, è stato utilizzato il Gradient Boosted Regression Tree (GBRT) per stimare i punteggi BPRS dei pazienti, segnando un importante progresso nel monitoraggio continuo. In seguito, vari studi hanno tentato di perfezionare questo approccio con tecniche più sofisticate. Ad esempio, in [11] è stato sviluppato un modello di deep learning chiamato *RelapsePredNet*, basato su Long Short-Term Memory (LSTM). L'architettura prevedeva l'inserimento delle caratteristiche in un layer LSTM bidirezionale, che apprendeva la differenza tra ricaduta e non ricaduta. Le attivazioni venivano poi passate a due livelli *fully connected* per ottenere la predizione finale. Questo modello è stato anche combinato con il *ClusterRFModel*, un modello di Random Forest che utilizza tecniche di clustering. Altri approcci come quello in [12], hanno utilizzato modelli di clustering come Gaussian Mixture Model (GMM) e Partition Around Medoids (PAM) per caratterizzare i comportamenti basati sui dati dei sensori. Le rappresentazioni ottenute sono state utilizzate per addestrare un modello di previsione delle ricadute con Balanced Random Forest, personalizzando le caratteristiche in base ai dati di pazienti simili. In [13] viene valutato l'utilizzo di classificatori più semplici come il Naive Bayes, adatto a dataset più piccoli e sbilanciati. In aggiunta a quest'ultimo, sono stati valutati classificatori come il Balanced Random Forest (BRF) e l'EasyEnsemble (EE), i quali hanno dimostrato di essere adatti per l'apprendimento su dataset squilibrati. Nonostante i progressi, tutti questi modelli condividono ancora un problema comune: l'accuratezza delle predizioni, seppur promettente, rimane insufficiente per l'uso clinico principalmente a causa della scarsità di dati disponibili.

CAPITOLO 3

TECNOLOGIE UTILIZZATE

In questo capitolo verrà analizzato lo stack tecnologico scelto per la realizzazione del sistema, con un focus specifico sui framework adottati, sui linguaggi di programmazione utilizzati e sugli strumenti impiegati per la gestione dei dati. Infine, verrà fatta una breve panoramica sull'ambiente di sviluppo utilizzato.

3.1 Flutter

Flutter è un framework open-source sviluppato da Google per la creazione di applicazioni multipiattaforma, che permette di sviluppare app native per Android, iOS, web e desktop utilizzando un unico codice sorgente. Basato sul linguaggio di programmazione Dart, la caratteristica principale di Flutter è il rendering nativo dell'interfaccia utente: invece di utilizzare i componenti di sistema delle diverse piattaforme, Flutter disegna tutto tramite il proprio motore grafico, garantendo così un controllo totale sull'aspetto grafico dell'app e prestazioni vicine a quelle di un'app nativa. La costruzione dell'interfaccia avviene tramite widget, che rappresentano

componenti altamente personalizzabili, ideali per creare interfacce moderne e dinamiche.

```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: const Text('Hello World App'),
      ),
      body : const SafeArea(
        child:  Center(
            child: Text('Hello, World!'),
          ),
      ),
      )
    );
```

Figura 3.1: Esempio di Widget in Flutter

Infine, Flutter è supportato da una vasta community di sviluppatori e offre numerosi pacchetti e plugin, che permettono di estendere facilmente le funzionalità delle applicazioni. Tutte queste caratteristiche hanno portato alla scelta di Flutter per lo sviluppo dell'applicazione mobile.

3.2 CARP Mobile Sensing

La Copenhagen Research Platform è un progetto open-source gestito da ricercatori della Sezione Digitale della Salute e del Dipartimento di Tecnologia della Salute presso la Technical University of Denmark. La progettazione e l'implementazione di CARP sono iniziate nel 2017 presso il Copenhagen Center for Health Technology (CACHET) come parte del progetto "Biometric Healthcare Research Platform" (BHRP), ma da allora è stata utilizzata in molti studi e applicazioni di CACHET. CARP è progettato per due scopi principali: è una piattaforma per il digital phenotyping, che raccoglie dati longitudinali e in tempo reale da partecipanti, e un'architettura software open-source per sviluppare applicazioni per la salute mobile (mHealth)

specifiche per malattie, come terapie comportamentali e analisi di dati cardiovascolari e diabetici.

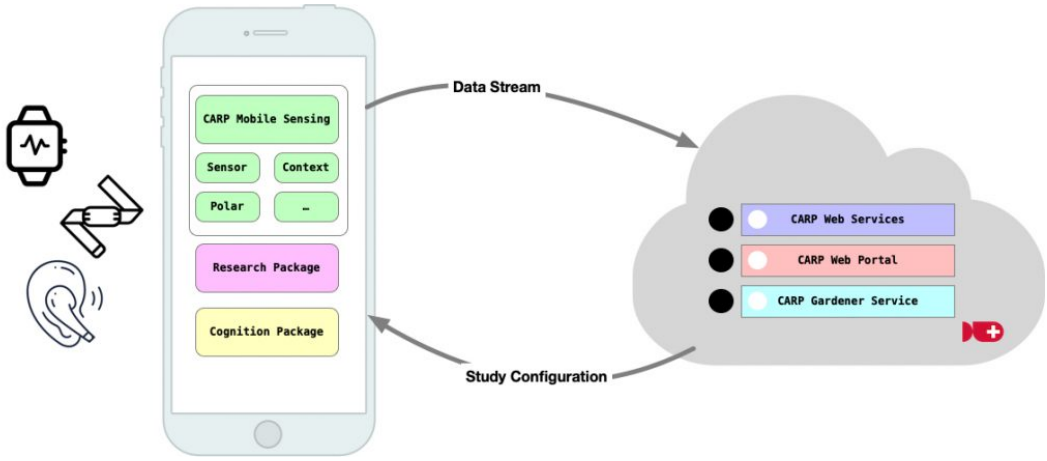

Figura 3.2: Panoramica della famiglia di componenti CARP

Una delle componenti di CARP è CARP Mobile Sensing (CAMS) [14], un package Flutter progettato per raccogliere dati sensoriali dai sensori di degli smartphone e dai dispositivi indossabili collegati allo smartphone. CAMS offre un'API per lo sviluppo di app cross-platform su Android e iOS con codice sorgente condiviso ed è progettata per essere altamente estendibile, in modo da permettere l'integrazione di nuovi metodi e trasformatori, con l'obiettivo di consentire la creazione di app mHealth personalizzate per la raccolta e l'uso di dati sulla salute dei pazienti.

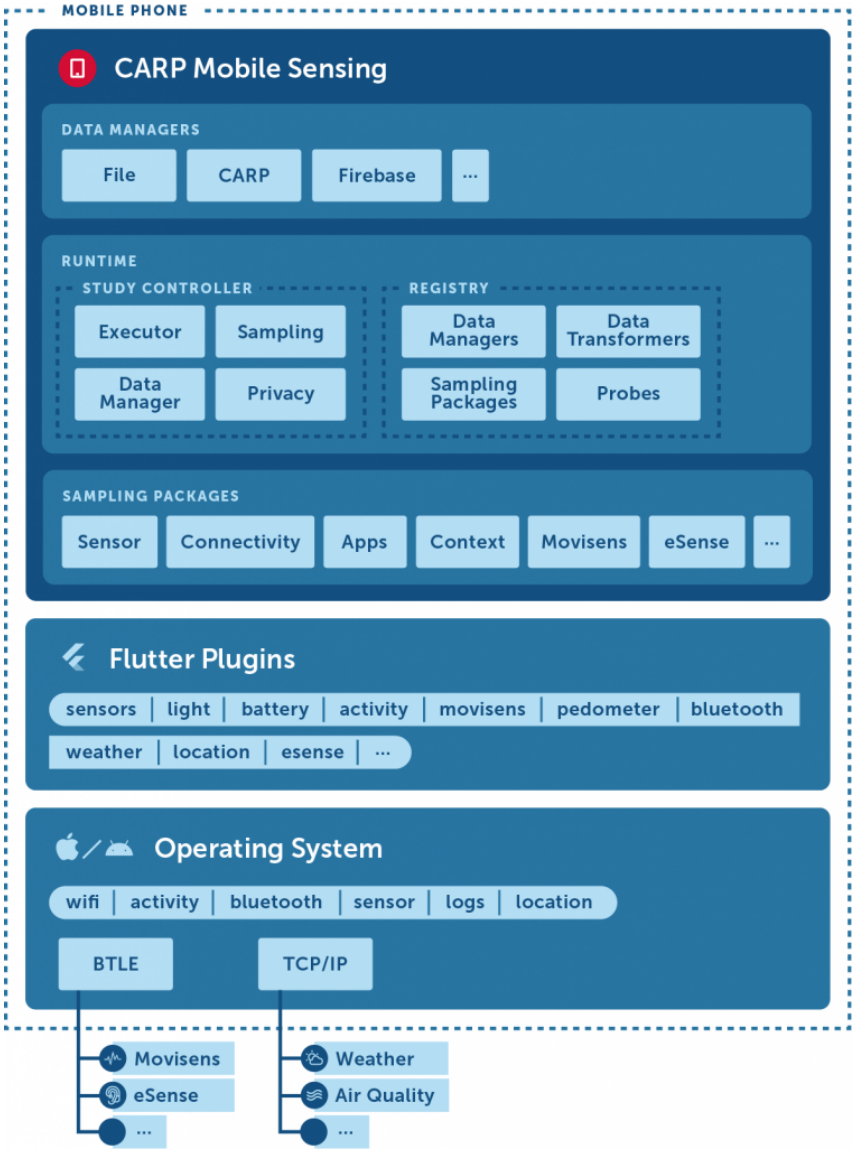

Figura 3.3: L'architettura di CAMS

3.3 Python

Python è un linguaggio di programmazione ad alto livello, creato da Guido van Rossum e rilasciato per la prima volta nel 1991. È noto per la sua semplicità e leggibilità, grazie a una sintassi chiara e pulita che lo rende facile da imparare e da utilizzare, rendendolo una scelta ideale sia per principianti che per sviluppatori esperti. Python supporta vari paradigmi di programmazione, inclusi quello orientato agli oggetti, quello procedurale e quello funzionale,

rendendolo molto versatile e adatto a una vasta gamma di applicazioni. Uno dei punti di forza di Python è la sua vasta libreria standard, che fornisce moduli e funzioni per la gestione di file, comunicazione di rete, interfaccia utente e molto altro, riducendo così la necessità di riscrivere codice complesso. Inoltre, la sua comunità open-source ha sviluppato un'ampia gamma di librerie esterne per compiti specifici, come lo sviluppo web, il machine learning, l'intelligenza artificiale e la visualizzazione dei dati. Python è stato utilizzato per l'elaborazione e la visualizzazione dei dati grezzi ricevuti dall'applicazione mobile, oltre che per la creazione del modello di predizione.

3.4 Flask

Flask è un microframework per lo sviluppo web in Python, creato da Armin Ronacher e rilasciato nel 2010. È noto per la sua semplicità, leggerezza e flessibilità, rendendolo adatto a progetti di diverse dimensioni, dai prototipi a applicazioni più complesse. La sua struttura minimalista consente di partire da una base semplice e aggiungere solo i componenti necessari, con un sistema di routing intuitivo per definire URL e relative funzioni. Flask offre un'ampia possibilità di estensione, grazie alla compatibilità con librerie e strumenti per la gestione dei database, l'autenticazione degli utenti e altro ancora. Inoltre, è sostenuto da una community attiva che fornisce numerosi pacchetti e estensioni. Questa combinazione di semplicità e versatilità ha reso Flask uno dei framework web più popolari nel panorama dello sviluppo software.

```
1   from flask import Flask
2
3   app = Flask(__name__)
4
5
6   @app.route('/')
7   def dashboard():
8       return 'Hello World!'
9
10
11  if __name__ == '__main__':
12      app.run()
```

Figura 3.4: Esempio di Hello World in Flask

3.5 TensorFlow

TensorFlow è una libreria open-source sviluppata da Google per il machine learning e il deep learning, molto utilizzata per la creazione di modelli di intelligenza artificiale. La sua architettura flessibile consente di costruire e addestrare reti neurali complesse, facilitando la gestione di dati di grandi dimensioni e l'implementazione di algoritmi avanzati. Grazie a TensorFlow, è possibile sfruttare l'accelerazione hardware, come GPU e TPU, per migliorare le prestazioni di addestramento, rendendolo ideale per applicazioni in tempo reale e progetti di ricerca. TensorFlow è stata la libreria principale utilizzata per lo sviluppo del modello di predizione.

3.6 Scikit-learn

Scikit-learn è una libreria Python per il machine learning che fornisce una vasta gamma di strumenti per l'analisi dei dati e la costruzione di modelli predittivi. Essa offre implementazioni di algoritmi di classificazione, regressione e clustering, oltre a metodi per la pre-elaborazione dei dati e la valutazione dei modelli. La semplicità e l'intuitività di Scikit-learn, unite a una documentazione dettagliata, la rendono una scelta popolare tra i ricercatori e gli sviluppatori per progetti di data science e machine learning. Scikit-learn è stata utilizzata a supporto di TensorFlow per lo sviluppo del modello di predizione, in particolare per la pre-elaborazione dei dati.

3.7 MongoDB

MongoDB è un database NoSQL orientato ai documenti, lanciato nel 2009, che utilizza un modello di dati flessibile basato su documenti in formato BSON (Binary JSON). Questa struttura consente di gestire grandi volumi di dati non strutturati e semistrutturati in modo semplice e intuitivo. Una delle caratteristiche principali di MongoDB è la scalabilità orizzontale, che permette

di distribuire i dati su più server tramite sharding. Supporta ricerche complesse grazie a funzionalità avanzate di indicizzazione e querying, oltre a operazioni di aggregazione per analisi in tempo reale. La compatibilità con vari linguaggi di programmazione, come Python, Java e JavaScript, ne facilita l'integrazione in diverse applicazioni.

3.8 IntelliJ IDEA

IntelliJ IDEA è un ambiente di sviluppo integrato (IDE) creato da JetBrains, progettato per semplificare lo sviluppo di software, in particolare per il linguaggio Java. Rilasciato per la prima volta nel 2001, IntelliJ offre un'ampia gamma di funzionalità avanzate, come l'autocompletamento intelligente del codice, il refactoring automatico e un sistema di navigazione efficiente, che migliorano la produttività degli sviluppatori. L'IDE supporta anche vari linguaggi di programmazione, tra cui Kotlin, Groovy e Scala, grazie a un'architettura basata su plugin che consente di estendere facilmente le sue funzionalità. IntelliJ è noto per la sua integrazione con strumenti di version control come Git e per il supporto a framework e tecnologie moderne, come Spring e Java EE. Una delle caratteristiche distintive di IntelliJ IDEA è il suo sistema di analisi del codice, che fornisce suggerimenti in tempo reale per migliorare la qualità del codice e ridurre gli errori. Grazie a queste funzionalità, IntelliJ è diventato uno degli IDE più popolari nel panorama dello sviluppo software, apprezzato sia da sviluppatori principianti che esperti.

CAPITOLO 4

SOLUZIONE PROPOSTA

In questo capitolo verrà fornita una panoramica approfondita della soluzione proposta, evidenziando la struttura complessiva del sistema e le sue principali componenti. Verranno descritte nel dettaglio le interazioni tra l'applicazione mobile, il server e il modello di predizione, mostrando come i dati raccolti dai sensori del dispositivo mobile siano gestiti e utilizzati per il monitoraggio remoto dei pazienti. Seguirà un'analisi dei requisiti funzionali e dei requisiti non funzionali. Saranno esaminati gli obiettivi specifici del sistema. Infine, verranno illustrati i principali casi d'uso, accompagnati da un Class Diagram che rappresenta la struttura dei dati e un Deployment Diagram che mostrerà come le componenti del sistema sono distribuite e interconnesse.

4.1 Obiettivi del sistema

4.1.1 Obiettivo principale

L'obiettivo principale del sistema è quello di monitorare e raccogliere dati sui pazienti schizofrenici in modo continuativo e affidabile. Utilizzando sensori integrati nel telefono, il sistema registra informazioni su mobilità, interazione sociale e condizioni ambientali. Questo monitoraggio costante permette

interventi tempestivi e supporta i medici, migliorando la qualità della vita dei pazienti.

4.1.2 Obiettivi specifici

- **Ridurre le ricadute attraverso la previsione**: Implementando algoritmi predittivi basati sui dati raccolti, il sistema intende identificare schemi che possano indicare un possibile deterioramento della salute mentale. Questo approccio proattivo mira a intervenire tempestivamente per prevenire ricadute e migliorare la qualità della vita del paziente.
- **Facilitare la gestione remota dei pazienti da parte dei medici**: Il sistema deve fornire agli operatori sanitari strumenti efficaci per monitorare i pazienti a distanza. Questo include la possibilità di visualizzare le metriche raccolte e ricevere avvisi in caso di anomalie, migliorando così la capacità dei medici di intervenire quando necessario.
- **Calcolare metriche utili per lo studio clinico**: Oltre alla gestione dei pazienti, il sistema si propone di generare metriche cliniche significative che possano contribuire alla ricerca. Queste metriche possono includere la distanza percorsa, la durata delle conversazioni e l'ampiezza audio, fornendo dati preziosi per studi futuri e per migliorare le pratiche cliniche.

4.2 Requisiti del sistema

In questa sezione verranno definiti in modo chiaro e dettagliato le caratteristiche che il sistema deve soddisfare per garantirne un funzionamento corretto ed efficiente. Saranno illustrati sia i requisiti funzionali, che descrivono i comportamenti specifici e le funzionalità che il sistema deve offrire, sia i requisiti non funzionali, che comprendono aspetti di performance, sicurezza, usabilità e affidabilità. Inoltre, saranno discusse le principali metriche da calcolare per valutare l'efficacia del monitoraggio dei pazienti e del modello

predittivo. Questa sezione servirà da base per guidare lo sviluppo del sistema e garantirne la coerenza con gli obiettivi progettuali.

4.2.1 Requisiti funzionali

I requisiti funzionali delineano le principali operazioni e comportamenti che il sistema deve garantire per rispondere agli obiettivi. Di seguito sono elencate le funzionalità principali:

- **Raccolta dati dai sensori del dispositivo mobile**: Il sistema deve essere in grado di acquisire dati dai sensori del telefono, tra cui:
  - Coordinate geografiche e velocità di spostamento, utili per monitorare il livello di attività fisica.
  - Accensione/spegnimento dello schermo, utile per calcolare il tempo di interazione con il dispositivo.
  - Livello di luminosità ambientale, per valutare le condizioni dell'ambiente in cui si trova il paziente.
  - Dati dell'accelerometro relativi ai movimenti del dispositivo, per analizzare i movimenti e le attività quotidiane.
  - Rumore ambientale, utile per analizzare il livello sonoro e la qualità delle conversazioni.
  - Conversazioni, per valutare l'attività sociale.

  Queste misure verranno poi elaborate per fornire informazioni utili alla predizione delle ricadute nei pazienti schizofrenici.

- **Archiviazione e gestione dei dati**: I dati raccolti dai sensori devono essere salvati in modo sicuro nel database, rispettando la struttura prevista e includendo informazioni temporali precise.

- **Elaborazione delle metriche**: Il sistema deve calcolare specifiche metriche basate sui dati raccolti. Tali metriche devono essere calcolate su base oraria e includono:

  - Distanza percorsa a partire dai dati GPS, espressa in kilometri (km).
  - Numero di luoghi significativi visitati dal paziente a partire dai dati GPS.
  - Durata delle conversazioni e ampiezza audio media tramite il microfono, entrambi espressi in decibel (db).
  - Cambiamenti posturali attraverso l'accelerometro, espressi in metri al secondo quadrato (m/s^2).
  - Utilizzo totale del telefono tramite accensione/spegnimento dello schermo, espresso in minuti (min).
  - Numero di utilizzi del telefono, tramite accensione/spegnimento dello schermo.
  - Ampiezza media dell'esposizione alla luce, espressa in lux (lux).

  L'obiettivo è fornire un quadro dettagliato del comportamento del paziente su base oraria, utile per monitorare il suo stato psicologico e fisico.

- **Interfaccia per il monitoraggio**: Il sistema deve offrire un'interfaccia per visualizzare le metriche raccolte e le elaborazioni eseguite, fornendo un accesso rapido ai dati di ogni paziente.
- **Integrazione con il modello predittivo**: Il sistema deve essere in grado di fornire i dati orari al modello predittivo, il quale elaborerà tali informazioni per stimare la probabilità di ricaduta del paziente.

4.2.2 Requisiti non funzionali

I requisiti non funzionali riguardano aspetti di qualità e performance del sistema:

- **Performance**: Il sistema deve garantire la raccolta dei dati in tempo reale, senza rallentamenti o perdita di informazioni. Inoltre, l'impatto

dell'applicazione sulla batteria del dispositivo mobile deve essere ridotto al minimo.

- **Affidabilità**: Il sistema deve garantire la continuità della raccolta dei dati anche in caso di interruzioni improvvise, come la terminazione forzata dell'applicazione da parte del sistema Android.
- **Usabilità**: L'interfaccia deve essere intuitiva e semplice da usare, sia per i pazienti che per i medici.

4.3 Casi d'uso

In questa sezione verranno descritti i principali casi d'uso del sistema, distinti tra i due attori principali: Medico e Paziente. Saranno evidenziate le interazioni e le funzionalità disponibili per ciascun attore: il Medico potrà monitorare da remoto i pazienti, consultare i dati raccolti e analizzare i risultati predittivi per intervenire tempestivamente in caso di rischio di ricaduta; il Paziente, invece, parteciperà passivamente al monitoraggio continuo tramite i sensori del dispositivo mobile. Verranno illustrati anche i flussi che garantiscono un'integrazione fluida tra raccolta dati, elaborazione e consultazione.

4.3.1 Casi d'uso relativi al Paziente

Identificativo	UC_P_01
Nome	Rilevazione dei dati
Descrizione	Il Paziente avvia l'applicazione ed inizia il rilevamento dei dati
Attore Principale	Paziente
Attori secondari	NA
Condizione di ingresso	Il Paziente apre l'applicazione

Condizione di uscita in caso di successo	Inizia il rilevamento dei dati e viene notificato il Paziente
Condizione di uscita in caso di fallimento	Il rilevamento dei dati non inizia
Flusso di eventi principale	1. Il Paziente apre l'applicazione 2. Il Sistema avvia la rilevazione dati e mostra una pagina di caricamento 3. Il Sistema conclude l'avvio della rilevazione dei dati e mostra la schermata principale 4. Il Sistema notifica il paziente che sta monitorando i suoi dati
Flusso di eventi di errore: Errore nell'avvio	2.1 Si verifica un errore nell'avvio della rilevazione dei dati 2.2 Il Sistema rimane bloccato nella fase di caricamento
Flusso di eventi di errore: Terminazione forzata	1.1 Scompare la notifica del Sistema per l'utente relativa al monitoraggio dei dati a causa di una chiusura forzata da parte del sistema Android 1.2 L'utente riavvia l'applicazione

Tabella 4.1: Caso d'uso UC_P_01

4.3.2 Casi d'uso relativi al Medico

Identificativo	UC_M_01
Nome	Consulazione dati paziente
Descrizione	Il medico effettua l'accesso alla piattaforma e consulta i dati del paziente
Attore Principale	Medico

Attori secondari	NA
Condizione di ingresso	Il Medico accede alla piattaforma
Condizione di uscita in caso di successo	Il Medico consulta i dati del paziente
Condizione di uscita in caso di fallimento	Il Medico non riesce ad accedere alla piattaforma
Flusso di eventi principale	1. Il Medico inserisce le credenziali nel form di autenticazione 2. Il Sistema controlla che le credenziali siano corrette 3. Il Sistema effettua l'accesso all'area personale del Medico 4. Il Medico seleziona il paziente di cui vuole consultare i dati 5. Il Sistema carica i dati del paziente richiesto dal Medico 6. Il Medico consulta i dati
Flusso di eventi di errore: Autenticazione fallita	2.1 Il Sistema notifica al Medico che le credenziali inserite non sono corrette e lo invita a reinserirle

Tabella 4.2: Caso d'uso UC_M_01

4.4 Modello dei dati

In questa sezione, descriveremo le principali classi utilizzate nel sistema, con un focus sulle entità fondamentali e le loro relazioni.

4.4.1 Paziente

Descrizione

Un paziente monitorato dal sistema.

Attributi

- **ID**: Un identificatore univoco per il paziente.
- **Nome**: Il nome del paziente.
- **Cognome**: Il cognome del paziente.
- **Data di nascita**: La data di nascita del paziente.
- **Sesso**: Il sesso biologico del paziente.
- **Indirizzo**: L'indirizzo di residenza del paziente.
- **Contatti**: Una serie di recapiti telefonici relativi sia al paziente che ai suoi parenti o custodi.

Relazioni

- Un Paziente ha una relazione uno-a-molti con Dati Sensore, dato che un paziente genera molti dati tramite sensori nel corso del tempo.
- Un Paziente ha una relazione molti-a-molti con Medico, in quanto più pazienti possono essere supervisionati da più medici.
- Un Paziente ha una relazione di uno-a-uno con Dati Predizione, dato che un paziente ha associata una predizione.

4.4.2 Medico

Descrizione

Il personale sanitario che supervisiona i pazienti e analizza i dati raccolti dal sistema di monitoraggio.

Attributi

- **ID**: Un identificatore univoco per il medico.
- **Nome**: Il nome del medico.
- **Cognome**: Il cognome del cognome.
- **Username**: L'username per l'accesso alla piattaforma.
- **Password**: La password per l'accesso alla piattaforma.

Relazioni

- Un Medico ha una relazione di molti-a-molti con un Paziente, in quanto più medici possono supervisionare più pazienti.
- Un Medico ha una relazione di molti-a-molti con i Dati Elaborati, dato che più medici possono consultare i dati.
- Un Medico ha una relazione di molti-a-molti con Dati Predizione, dato che più medici possono consultare più predizioni.

4.4.3 Dati Sensore

Descrizione

Le rilevazioni effettuate dai vari sensori.

Attributi

- **ID**: Un identificatore unico per la rilevazione.
- **Tipo**: Il tipo di sensore che ha generato il dato.
- **Timestamp**: L'istante temporale in cui è stato raccolto il dato.
- **Valore**: Il valore registrato dal sensore.

Relazioni

- Un Dato Sensore ha una relazione molti-a-uno con Paziente, dato che più rilevazioni possono essere generate da un singolo paziente.
- Un Dato Sensore ha una relazione di molti-a-uno con con Dati Elaborati, in quanto più dati rilevati dai sensori possono essere usati per calcolare una singola metrica.

4.4.4 Dati Elaborati

Descrizione

Le metriche calcolate a partire dai dati grezzi.

Attributi

- **ID**: Un identificatore unico per la metrica.
- **Tipo**: Il tipo di metrica.
- **Data**: La data a cui fa riferimento la metrica.
- **Ora**: L'ora a cui fa riferimento la metrica.
- **Valore**: Il valore della metrica.

Relazioni

- Un Dato Elaborato ha una relazione di uno-a-molti con Dati Sensore, in quanto vengono utilizzati più dati rilevati dai sensori per il calcolo di una singola metrica.
- Un Dato Elaborato ha una relazione di molti-a-molti con Medico, dato che più medici possono consultare più metriche.
- Un Dato Elaborato ha una relazione di molti-a-uno con Dati Predizione, in quanto più metriche vengono utilizzate per una singola predizione.

4.4.5 Dati Predizione

Descrizione

I dati relativi alle predizioni di ricaduta di un paziente.

Attributi

- **ID**: L'identificativo del paziente a cui fa riferimento la predizione.
- **Campioni analizzati**: Il numero di campioni analizzati per la predizione.
- **Giorni anomali**: Il numero di giorni anomali rilevati.
- **Rischio ricaduta**: Indica se il paziente è a rischio ricaduta.

Relazioni

- Un Dato Predizione ha una relazione di uno-a-molti con Dati Elaborati, in quanto più metriche vengono utilizzati per effettuare una predizione.
- Un Dato Predizione ha una relazione di molti-a-molti con Medico, in quanto più medici possono visualizzare più predizioni.
- Un Dato Predizione ha una relazione di uno-a-uno con Paziente, dato che una predizione è associata ad un solo paziente.

4.4.6 Class diagram

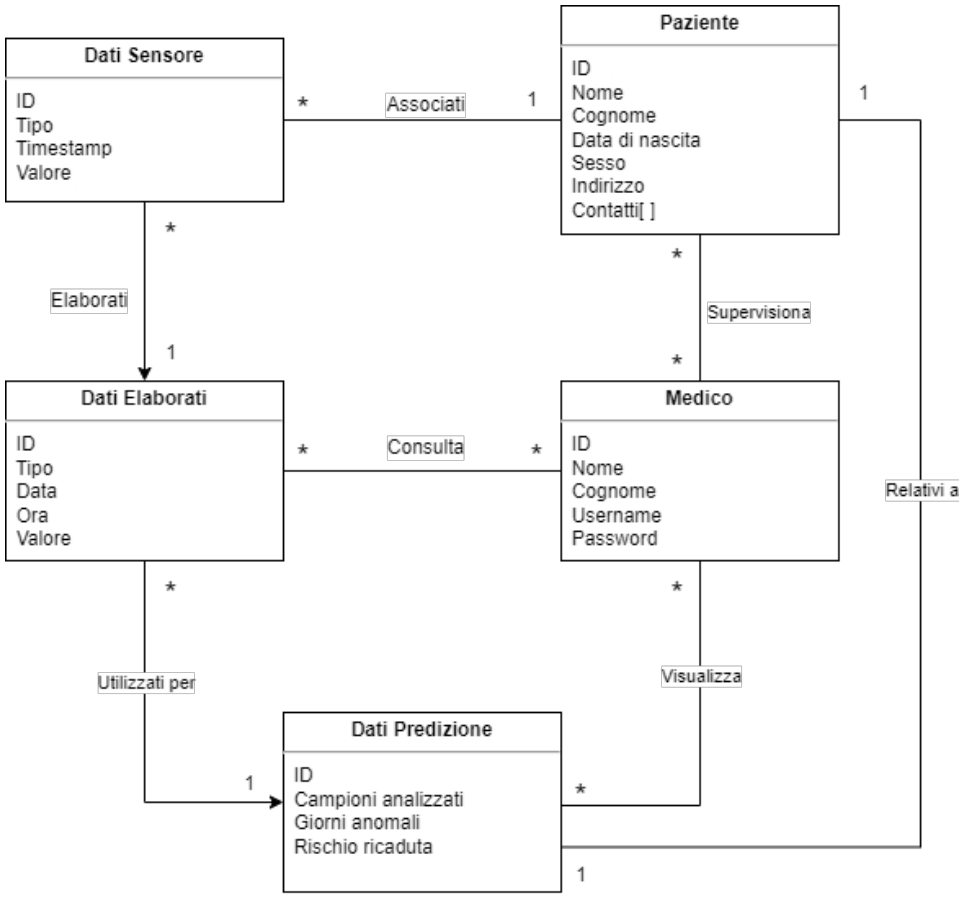

Figura 4.1: Class diagram per il modello dei dati

4.5 Architettura del sistema

L'interazione tra l'applicazione mobile, il server e il modello di predizione segue un flusso di lavoro ben definito, che consente di raccogliere, elaborare e analizzare i dati dei pazienti in modo efficiente.

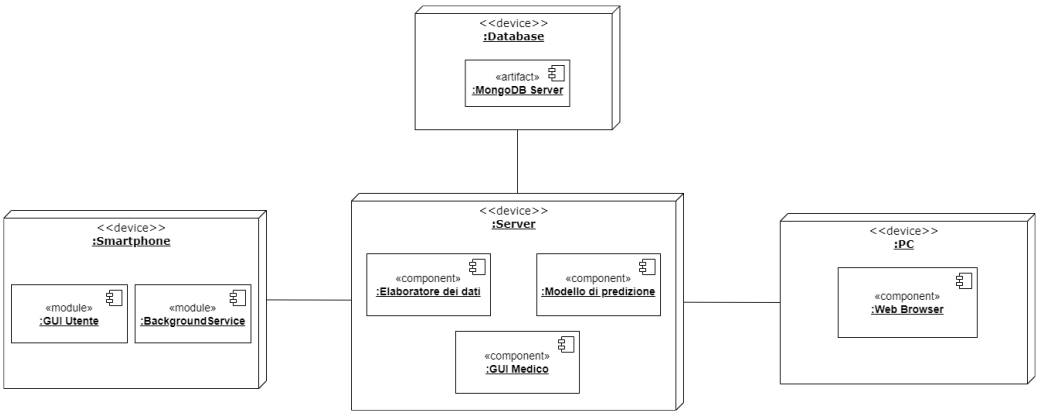

Figura 4.2: Deployment diagram per il sistema proposto

L'applicazione mobile è il punto di raccolta dei dati, che monitora continuamente il paziente utilizzando i sensori del dispositivo (come accelerometri per il movimento, GPS per la posizione, ecc.). Questi dati vengono raccolti in tempo reale e trasmessi periodicamente al server tramite connessioni sicure. Il server agisce come il cuore del sistema: riceve i dati dall'applicazione mobile, li elabora e li archivia in un database. Qui vengono effettuate le operazioni di pulizia e trasformazione dei dati per garantire che siano in un formato coerente e pronto per l'analisi. Il server è anche responsabile di gestire i meccanismi di sicurezza, come la crittografia, per proteggere la privacy del paziente. Una volta che i dati sono stati elaborati, il server li invia al modello di predizione basato su intelligenza artificiale. Il modello, addestrato su dati storici di pazienti, utilizza algoritmi di machine learning per analizzare i nuovi dati e identificare pattern o segnali che potrebbero indicare una ricaduta imminente. Questo processo previsionale si basa su parametri come i cambiamenti nel comportamento, la variazione dell'attività fisica o il peggioramento dei sintomi rilevati dai sensori. Infine, i risultati del modello predittivo vengono trasmessi nuovamente al server, che li archivia e li rende disponibili per la consultazione da parte dei medici. I medici possono visualizzare i risultati tramite un'interfaccia web o ricevere notifiche in caso di rischio elevato di ricaduta, consentendo così di intervenire in modo tempestivo.

CAPITOLO 5

MODELLO DI PREDIZIONE

In questo capitolo verrà esaminato il modello di predizione implementato per la previsione delle ricadute. Verranno trattati diversi aspetti cruciali del processo, a partire dal dataset utilizzato per l'addestramento del modello, passando per il preprocessing applicato ai dati per migliorare la qualità dell'input. Saranno inoltre analizzate le feature selezionate, spiegando le motivazioni dietro la loro scelta. Infine, si discuterà dell'architettura del modello e delle tecniche adottate per il suo addestramento e validazione, con un focus sugli approcci impiegati per ottenere una buona performance.

5.1 Dataset

Il dataset scelto per la costruzione del modello di predizione è una versione de-identificata e approvata per uso pubblico dei dati raccolti nello studio CrossCheck [15], che comprende esclusivamente la parte relativa ai dati raccolti tramite smartphone. Questo dataset è costituito da tre file:

- **CrossCheck_Daily_Data.csv**: Contiene le caratteristiche comportamentali giornaliere.

- **CrossChec_Hourly_Data_v1_08212020.csv.gz**: Contiene le caratteristiche comportamentali orarie.
- **Eureka Data Dictionary Daily and Hourly v1 08212020.xlsx**: Offre una panoramica delle colonne presenti nei due file precedenti.

Questi dati sono gli stessi utilizzati nello studio originale di CrossCheck [5], quindi rappresentano misurazioni comportamentali dei pazienti monitorati tramite sensori degli smartphone imputate nell'arco dell'intera giornata ed in periodi di tempo di 6 ore ciascuno. Tuttavia, essendo una versione de-identificata e limitata ai soli dati dello smartphone, il dataset non include le etichette che distinguono tra ricaduta e non-ricaduta per ciascun paziente. Questa assenza di label complica il compito di addestrare un modello supervisionato per la previsione delle ricadute, rendendo necessario l'uso di tecniche di pre-processing o l'acquisizione di informazioni aggiuntive per identificare tali eventi. Oltre a ciò, il file contenente le caratteristiche comportamentali orarie è privo di una colonna che consenta di identificare in modo univoco l'appartenenza di un dato ad uno specifico paziente. Questa mancanza rende impossibile collegare i dati orari a quelli giornalieri, impedendo l'aggregazione dei dati tra i due livelli temporali. Per questo motivo, solo il file contenente le caratteristiche comportamentali giornaliere sarà utilizzato per l'addestramento del modello.

5.2 Preprocessing dei dati

Per prima cosa, ci si è occupati di gestire le righe contenenti valori `NaN`. Tralasciando la colonna `ema_score`, che non è stata selezionata come feature, le colonne relative alla distanza percorsa e al numero di luoghi visitati presentavano alcune righe con valori `NaN`. Data la dimensione ridotta del dataset, è stato deciso di non eliminare queste righe, ma di sostituire i valori mancanti con 0. Successivamente, si è scelto di eliminare i pazienti con meno di 30 giorni di dati. Questo valore è stato determinato empiricamente, basandosi

su studi descritti nel Capitolo 2. Pazienti con meno di 30 giorni di dati avrebbero potuto introdurre rumore, peggiorando le performance del modello, e risulterebbe difficile fare previsioni affidabili con un numero così ridotto di dati. In seguito, si è proceduto con una piccola fase di feature extraction. Sebbene le colonne del dataset contenessero già dati elaborati, dalla colonna relativa alla data delle rilevazioni sono state estratte ulteriori feature:

- Anno.
- Mese.
- Giorno.
- Giorno della settimana.
- Giorni dall'inizio dello studio.
- Trasformazioni sinusoidali e cosinusoidali per mese e giorno della settimana.

Le trasformazioni sinusoidali e cosinusoidali sono state applicate per modellare la ciclicità naturale dei mesi e dei giorni della settimana, evitando discontinuità numeriche (ad esempio tra dicembre e gennaio o tra domenica e lunedì). In questo modo, il modello riesce a cogliere meglio i pattern ciclici. Per quanto riguarda il mese, la formula per calcolare la sua trasformazione sinusoidale è

$$month\_sin = \sin\left(\frac{2\pi \cdot month}{12}\right)$$

mentre per la trasformazione cosinusoidale la formula è la stessa applicando il coseno (cos). Per il giorno della settimana, invece, la formula per il calcolo della trasformazione sinusoidale è

$$weekday\_sin = \sin\left(\frac{2\pi \cdot weekday}{7}\right)$$

anche in questo caso, la formula per la trasformazione cosinusoidale è la stessa, ma applicando il coseno (cos). Infine, i dati sono stati normalizzati utilizzando un MinMaxScaler. Questa tecnica riduce i valori delle feature all'interno di

un intervallo compreso tra 0 e 1, migliorando la performance del modello e assicurando che tutte le feature abbiano lo stesso peso durante l'addestramento. La formula applicata per la scalatura è la seguente:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

dove x è il valore originale della feature mentre x_{min} e x_{max} sono, rispettivamente, il valore minimo e massimo della feature nel dataset. Questo approccio assicura che tutte le feature, indipendentemente dalla loro scala originale, vengano trattate uniformemente.

5.3 Selezione delle feature

Le feature selezionate comprendono i dati elaborati dal resto del sistema e le feature estratte a partire dalla data descritte nella sezione precedente. Sfortunatamente, non è stato possibile utilizzare le misure relative all'accelerometro in quanto la misura era solamente presente nel file contenente le misurazioni orarie, che abbiamo già detto essere incompatibile con le rilevazioni giornaliere a causa della mancanza di un identificativo per il paziente. Di ogni feature è stata utilizzata la misura relativa all'intera giornata. Le feature selezionate sono:

- **eureka_id**: Identificativo univoco per ogni paziente nel dataset.
- **year**: Anno in cui sono stati raccolti i dati.
- **month**: Mese dell'anno (numerico).
- **day_of_month**: Giorno del mese in cui i dati sono stati raccolti.
- **weekday**: Giorno della settimana (lunedì=0, domenica=6).
- **days_since_start**: Giorni trascorsi dall'inizio dello studio.
- **month_sin**: Trasformazione sinusoidale del mese per rappresentare la ciclicità.

- **month_cos**: Trasformazione cosinusoidale del mese, complementare alla sinusoidale.
- **weekday_sin**: Trasformazione sinusoidale del giorno della settimana per rappresentare la ciclicità.
- **weekday_cos**: Trasformazione cosinusoidale del giorno della settimana, complementare alla sinusoidale.
- **audio_convo_duration_ep_0**: Durata totale delle conversazioni rilevate (in secondi).
- **audio_convo_num_ep_0**: Numero di conversazioni rilevate.
- **audio_amp_mean_ep_0**: Ampiezza audio media delle convesazioni rilevate.
- **light_mean_ep_0**: Deviazione standard della luce.
- **loc_dist_ep_0**: Distanza percorsa giornaliera (in metri).
- **loc_visit_num_ep_0**: Numero di luoghi visitati.
- **unlock_num_ep_0**: Numero di volte che il telefono è stato sbloccato.
- **unlock_duration_ep_0**: Durata totale in cui il telefono è stato sbloccato (in secondi).

5.4 Architettura del modello

Data la natura non etichettata dei dati, l'addestramento di un modello supervisionato per prevedere le ricadute risulta complicato. Pertanto, è stata adottata una strategia alternativa che consiste nella rilevazione delle anomalie nei pattern comportamentali dei pazienti, i quali possono servire come indicatori significativi delle ricadute. Per questo scopo, è stato scelto un autoencoder, noto per la sua efficacia nella rilevazione di anomalie nei dati non classificati. Gli autoencoder sono particolarmente efficaci per la

rilevazione di anomalie nei dati non classificati grazie alla loro capacità di apprendere rappresentazioni latenti delle feature. Durante il processo di addestramento, l'autoencoder cerca di ricostruire i dati originali attraverso un processo di compressione e decompressione. Le anomalie tendono a generare ricostruzioni di scarsa qualità, poiché non si conformano ai pattern appresi dal modello. Monitorando l'errore di ricostruzione, è possibile identificare osservazioni anomale, rendendo gli autoencoder strumenti potenti per l'analisi dei dati quando le etichette non sono disponibili. Il modello implementato è un autoencoder basato su LSTM, progettato per elaborare sequenze temporali e rilevare anomalie in dati non classificati. La prima parte del modello è rappresentata dall'input layer, che accetta sequenze di lunghezza variabile, definite da `window_size` e da `input_feature_dim`. Questa flessibilità è fondamentale per trattare dati che possono variare nel tempo. Successivamente, l'Encoder utilizza uno strato LSTM con 64 unità e funzione di attivazione ReLU. Questo strato comprime l'informazione della sequenza, mantenendo solo le caratteristiche rilevanti e riducendo la dimensionalità. Dopo l'encoding, la sequenza compressa viene replicata tramite RepeatVector, ripristinando la dimensione temporale. Successivamente, uno strato LSTM ricostruisce la sequenza originale, utilizzando un numero di unità pari a `input_feature_dim`. Lo strato di output utilizza un TimeDistributed con un layer denso (Dense layer) e una funzione di attivazione Sigmoid per produrre l'output ricostruito della sequenza originaria. Il modello è stato compilato utilizzando l'ottimizzatore Adam e la funzione di errore `mean_squared_error`, che confronta l'output ricostruito con l'input originale per misurare la precisione della ricostruzione. Questo approccio consente di identificare anomalie e comportamenti inattesi nei dati temporali, contribuendo a migliorare la capacità di previsione delle ricadute.

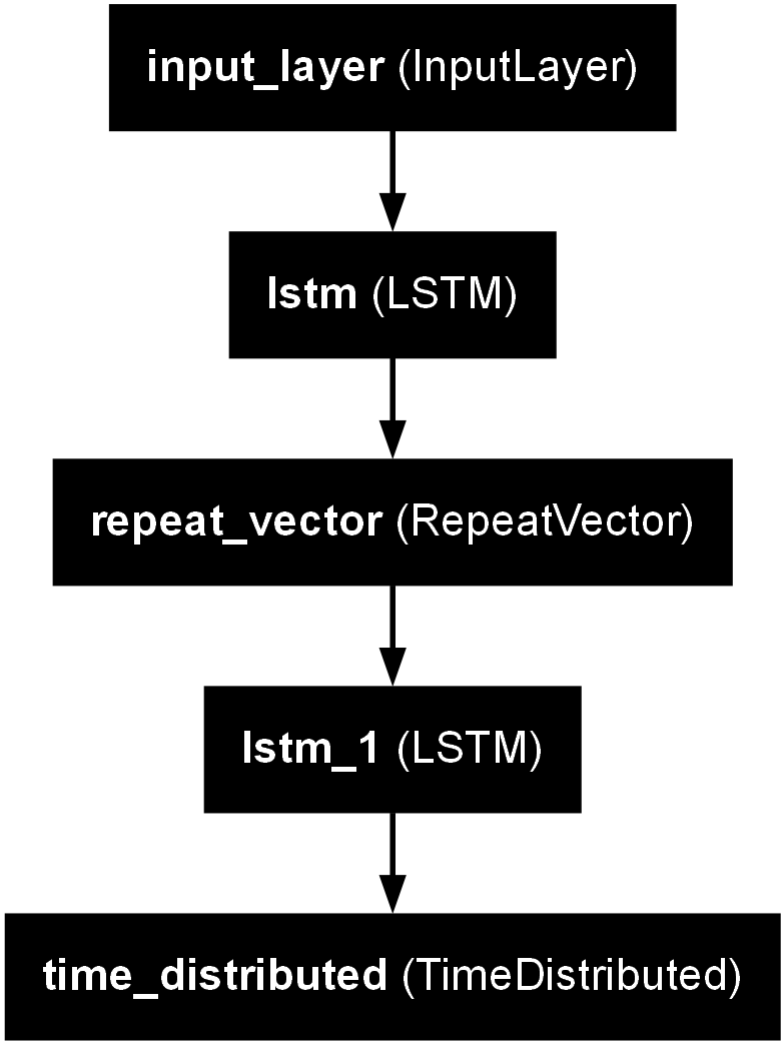

Figura 5.1: Schema dell'architettura del modello di predizione

5.5 Strategia di classificazione

Data l'assenza di etichette per una classificazione diretta, è emersa l'importanza di sviluppare un metodo di classificazione che si fondasse sull'analisi delle anomalie. Questo approccio è stato strutturato attorno a tecniche statistiche, con un focus particolare sull'imputazione attraverso percentili. In questo contesto, il 75° percentile è stato scelto come soglia critica per identificare i pazienti a rischio di ricaduta. Questa scelta non è stata casuale, ma frutto di un'analisi attenta, volta a differenziare i pazienti con comportamenti anomali da quelli che si collocano all'interno di una

gamma considerata normale. L'analisi dei giorni di anomalia ha permesso di identificare pazienti che superano questa soglia, classificandoli come a rischio di ricaduta. Utilizzando il 75° percentile come punto di riferimento, è possibile ottenere una misura oggettiva per il monitoraggio dei comportamenti dei pazienti, fornendo così una base per interventi tempestivi e mirati. Questa metodologia si basa sull'idea che, anche in assenza di etichette, è possibile utilizzare analisi statistiche per estrarre informazioni significative dai dati disponibili, contribuendo a migliorare la cura e il supporto forniti ai pazienti.

5.6 Addestramento e validazione

Il modello è stato addestrato e validato utilizzando la metodologia di cross-validation Leave-One-Patient-Out (LOPO). Questa strategia consente di addestrare il modello escludendo un singolo paziente alla volta, utilizzando i dati di tutti gli altri pazienti per l'addestramento. Questa pratica è fondamentale nel contesto clinico, in quanto permette di valutare l'efficacia del modello in modo più realistico e rappresentativo delle variazioni individuali presenti nei dati. La validazione tramite LOPO offre numerosi vantaggi. In primo luogo, assicura che il modello non venga influenzato da dati appartenenti allo stesso paziente durante la fase di addestramento, garantendo così una valutazione imparziale delle sue prestazioni. Inoltre, questa metodologia consente di analizzare la capacità del modello di generalizzare i risultati, identificando pattern comportamentali e anomalie che possono essere indicative di un rischio di ricaduta. Il processo di cross-validation non solo contribuisce a migliorare l'affidabilità delle previsioni, ma offre anche un'opportunità per ottimizzare le strategie di intervento, prendendo in considerazione le specificità di ogni paziente. In questo modo, è possibile affinare ulteriormente il modello e garantire che le sue applicazioni pratiche siano efficaci e pertinenti nel supporto ai pazienti. Il modello è stato addestrato con una batch size di 32 e per un totale di 50 epoche, bilanciando la necessità di un addestramento efficace con il rischio di overfitting.

CAPITOLO 6

IMPLEMENTAZIONE

In questo capitolo viene descritta l'implementazione del sistema di monitoraggio remoto per pazienti schizofrenici. Vengono illustrati i componenti principali dell'applicazione, tra cui la raccolta dati tramite sensori, l'architettura del server e l'integrazione del modello di predizione. Verranno inoltre discusse le tecnologie utilizzate, l'organizzazione del codice e le scelte progettuali fatte per garantire affidabilità, sicurezza e scalabilità del sistema.

6.1 Applicazione mobile

In questa sezione illustreremo l'implementazione dell'applicazione mobile, progettata per monitorare e raccogliere dati dai pazienti in tempo reale.

6.1.1 Architettura dell'applicazione

L'applicazione è strutturata seguendo il pattern architetturale BLoC (Business Logic Component), che separa la logica di business dalla presentazione dell'interfaccia utente in modo reattivo. La logica è centralizzata nei bloc, mentre l'interfaccia rimane leggera e focalizzata sulla visualizzazione dei dati, consentendo una gestione efficiente dei dati e facilitando la manutenibilità

dell'applicazione. Nello specifico, l'architettura dell'applicazione è organizzata nei seguenti quattro package:

- `sensing`: Gestisce l'acquisizione dei dati dai sensori del dispositivo. È responsabile della raccolta delle informazioni come la posizione GPS, i dati di movimento e altre metriche fisiologiche.
- `bloc`: Contiene i vari blocchi logici che implementano il pattern BLoC. Ogni bloc si occupa di gestire un flusso di dati specifico dell'applicazione, ricevendo eventi dall'interfaccia utente e producendo nuovi stati in base alla logica di business. Questi stati vengono poi inviati nuovamente all'interfaccia utente tramite stream reattivi.
- `model`: Contiene le classi che rappresentano i dati del dominio dell'applicazione. Queste classi definiscono la struttura dei dati e vengono utilizzate in tutta l'applicazione per manipolare le informazioni in modo coerente.
- `ui`: Gestisce l'interfaccia utente e contiene i widget di Flutter responsabili della visualizzazione dei dati. L'interfaccia utente riceve gli stati aggiornati dal bloc e li riflette nei componenti visivi, garantendo un'esperienza fluida e reattiva per l'utente.

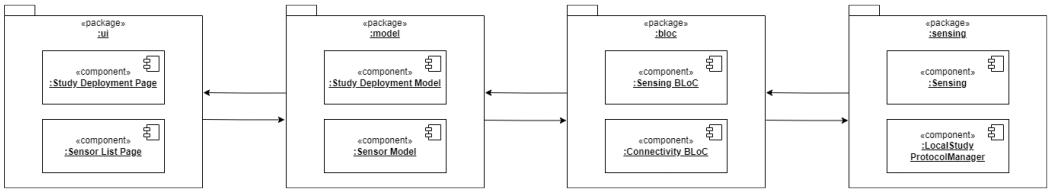

Figura 6.1: Interazione tra i package dell'applicazione

6.1.2 Package `sensing`

Il package `sensing` è responsabile della raccolta dei dati. È composto da due classi chiave: `LocalStudyProtocolManager` e `Sensing`.

`LocalStudyProtocolManager`

La classe `LocalStudyProtocolManager` configura un protocollo di studio locale per dispositivi mobili (`SmartphoneStudyProtocol`) definendo quali dati sensoriali devono essere raccolti, in che momento e con quale frequenza. Utilizza una combinazione di trigger e task per monitorare in modo continuo i dati provenienti da vari sensori dello smartphone.

```
// Define the online location service and add it as a 'connected device'
final locationService = LocationService();
protocol.addConnectedDevice(locationService, phone);

// Add a background task that continuously collects location
protocol.addTaskControl(
    ImmediateTrigger(),
    BackgroundTask(
      name: "Location",
      measures: [
        Measure(type: ContextSamplingPackage.LOCATION),
    ]),  // BackgroundTask
    locationService
);
```

Figura 6.2: Esempio di configurazione di un task

I task configurati nel protocollo sono stati progettati per soddisfare i requisiti di raccolta dati e includono le seguenti misure:

- Posizione GPS.
- Eventi legati all'accensione e spegnimento dello schermo.
- Livello di luce ambientale.
- Movimenti rilevati dall'accelerometro.
- Rumore ambientale.
- Registrazione audio dal microfono.

`Sensing`

La classe `Sensing` rappresenta il cuore del sistema di monitoraggio dei dati nel dispositivo mobile. Lavora come un singleton e può essere invocata tramite `Sensing()`, gestendo il flusso di dati raccolti dai vari sensori del dispositivo. Durante l'inizializzazione, la classe recupera il protocollo di studio locale, configura e avvia il controller dello studio responsabile del campionamento continuo dei dati. I dati raccolti vengono aggiunti a un buffer locale e, quando il dispositivo è connesso ad una rete Wi-Fi con accesso ad internet, vengono inviati periodicamente al server per l'elaborazione tramite una richiesta `http`. Oltre ad essere inviati al server, i dati sono salvati in locale in un file JSON per i dati numerici ed MP4 per i dati del microfono, per essere poi recuperati in caso di errori di connessione con il server e/o connessione internet assente per un periodo prolungato di tempo. Il flusso dei dati raccolti è accessibile tramite lo stream `measurements`, che fornisce le misure rilevate dai sensori in tempo reale.

6.1.3 Package `bloc`

Nel package bloc, due classi principali gestiscono la logica di business legata al sensing e all'invio dei dati al server: `SensingBLoC` e `ConnectivityBLoC`.

`SensingBLoC`

Questa classe è responsabile della gestione della logica di sensing, lavorando a stretto contatto con il livello di sensing implementato nella classe Sensing. Alcuni dei suoi compiti principali includono:

- **Gestione degli ID di studio e deployment**: Memorizza localmente lo stato del deployment di studio e del dispositivo tramite chiavi costanti (`STUDY_ID_KEY`, `STUDY_DEPLOYMENT_ID_KEY`, `DEVICE_ROLE_NAME_KEY`). Questi vengono memorizzati nelle preferenze dell'applicazione.

- **Avvio e arresto del sensing**: La classe fornisce metodi per avviare e fermare la raccolta dati attraverso `SmartPhoneClientManager`. Vengono utilizzate delle notifiche sullo smartphone per informare l'utente sullo stato di raccolta dei dati.
- **Ripresa automatica**: Supporta la ripresa automatica del sensing all'avvio dell'app, se configurata, anche in caso di terminazione forzata dell'applicazione da parte del sistema Android.
- **Modelli di studio e dispositivi**: Gestisce i modelli di studio e dispositivi attualmente attivi, inclusi i sensori attivi e quelli disponibili.

`ConnectivityBLoC`

La classe `ConnectivityBLoC` si occupa della gestione della connessione di rete e dell'invio periodico dei dati al server quando è connessa a una rete Wi-Fi. Le sue funzionalità includono:

- **Monitoraggio della connessione**: Utilizza il `plugin connectivity_plus` per rilevare cambiamenti di stato della connessione. Quando viene rilevata una connessione Wi-Fi, verifica se è disponibile l'accesso a internet.
- **Invio dati periodico**: Se connessa a una rete Wi-Fi con accesso a internet, avvia un timer per inviare i dati al server ogni 30 secondi. Se la connessione viene persa, interrompe l'invio dei dati.
- **Verifica accesso a Internet**: Prima di inviare i dati, controlla la presenza di internet effettuando una richiesta `http` al server.
- **Cancellazione e pulizia**: Quando l'oggetto viene eliminato, invia i dati rimanenti se possibile e interrompe la gestione del timer e della connessione.

La combinazione di queste due classi garantisce la raccolta continua dei dati dal dispositivo e il loro invio efficiente al server, solo quando è disponibile una connessione Wi-Fi con accesso a internet.

6.1.4 Package `model`

Il package `model` utilizza un `Model` per ciascuna pagina dell'interfaccia utente (UI). Ad esempio, il `Model StudyDeploymentModel` è associato al Widget UI `StudyDeploymentPage`. Il ruolo principale del `Model` è quello di fungere da ponte tra i dati e l'interfaccia utente, fornendo accesso sia ai dati in lettura che in scrittura (getters e setters). Questi dati, a loro volta, sono resi disponibili tramite il package `bloc`. Il `Model` si assicura che l'interfaccia utente abbia accesso ai dati aggiornati, reagendo ai cambiamenti e notificando il BLoC quando sono necessarie operazioni come la modifica dei dati o l'avvio di nuove attività di sensing. In questo modo, separa la logica di business dal codice che gestisce la presentazione, rendendo l'app più modulare, manutenibile e reattiva ai cambiamenti di stato.

6.1.5 Package `ui`

Il package `ui` è costituito dalle viste dell'interfaccia utente (UI), che rappresentano la parte visibile e interattiva per l'utente finale. Ogni vista UI è strettamente associata a un `Model` specifico, che viene passato nel costruttore del widget corrispondente. Questo modello di progettazione garantisce che ogni componente UI disponga dei dati necessari per svolgere la propria funzione. Ad esempio, lo stato del widget `StudyDeploymentPage` riceve un'istanza del `StudyDeploymentModel` al momento della sua creazione. Tale meccanismo consente al `Model` di essere disponibile in tutto il Widget, rendendo i dati facilmente accessibili per la visualizzazione o l'interazione. In questo modo, è possibile accedere ai dati gestiti dal `Model` in ogni punto della vista e utilizzarli per aggiornare l'interfaccia in tempo reale. Per esempio, per visualizzare il titolo dello studio e l'immagine corrispondente, è sufficiente fare riferimento ai dati contenuti nel `Model`, semplificando il flusso di informazioni e garantendo una separazione chiara tra logica di business e presentazione grafica. Questo approccio non solo rende il codice più leggibile e manutenibile, ma assicura

anche una migliore gestione dello stato e del ciclo di vita dei dati, contribuendo a un'interfaccia utente più reattiva e dinamica.

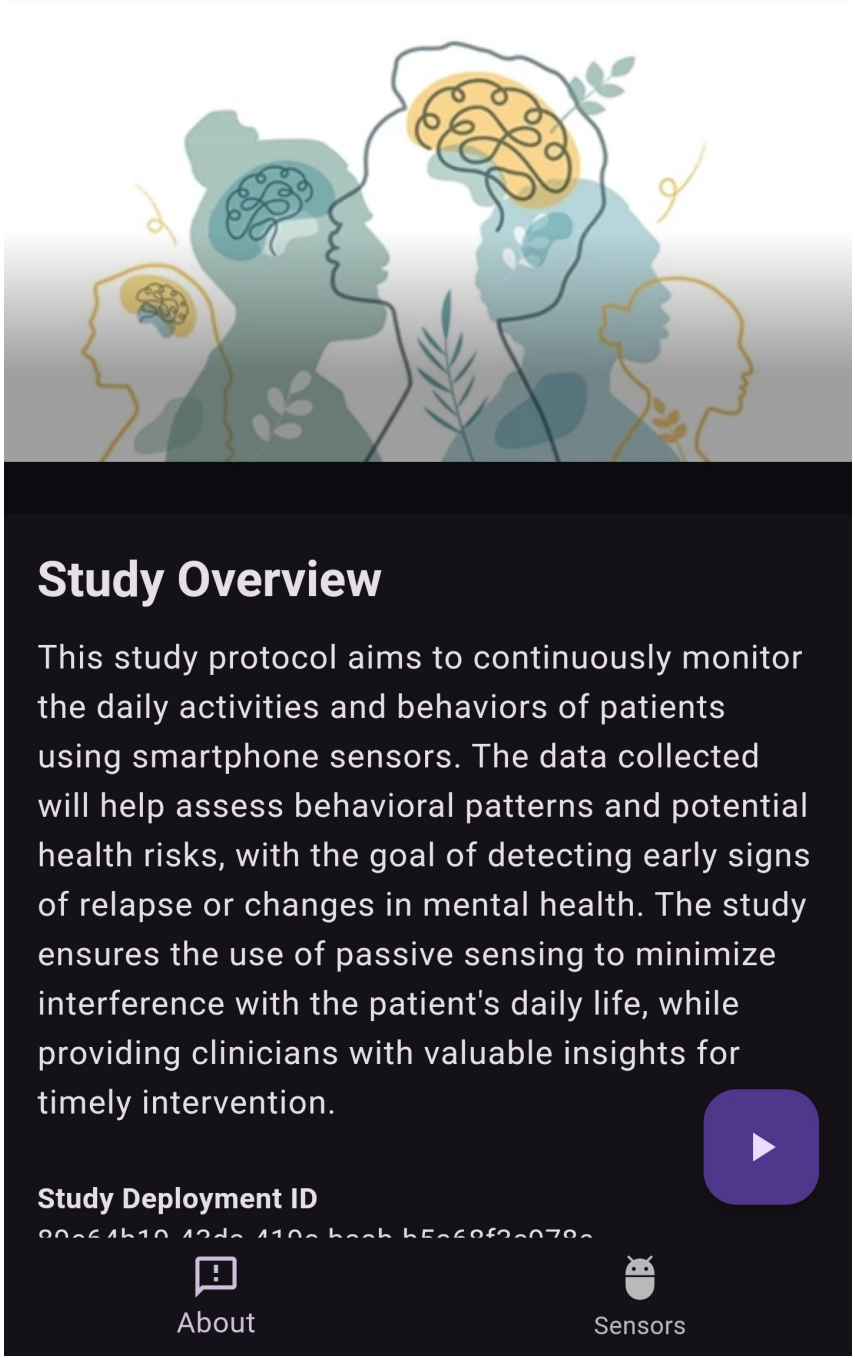

Figura 6.3: Schermata principale dell'applicazione

6.2 Server backend

In questa sezione descriveremo l'implementazione del server backend, evidenziando il suo ruolo cruciale nella comunicazione con l'applicazione mobile e nell'elaborazione dei dati ricevuti da quest'ultima.

6.2.1 Panoramica generale

Il server backend del sistema è stato sviluppato utilizzando Flask e si occupa di gestire l'interazione tra i dati raccolti dai sensori del telefono dei pazienti e la dashboard accessibile dai medici. Le sue principali funzionalità sono:

- **Ricezione e conservazione dei dati**: Il server riceve periodicamente i dati raccolti dai sensori presenti sul dispositivo mobile del paziente. Questi dati vengono poi memorizzati in un database per essere successivamente elaborati.
- **Elaborazione delle metriche orarie**: Ogni giorno il server elabora i dati raccolti dai sensori, calcolando le metriche orarie relative all'attività del paziente. Queste metriche includono informazioni utili per il monitoraggio delle condizioni del paziente.
- **Dashboard per i medici**: Il server fornisce una dashboard accessibile tramite autenticazione, che consente ai medici di visualizzare i dati e le metriche calcolate.

6.2.2 Ricezione e conservazione dei dati

Una delle principali funzionalità del server è la raccolta continua e automatizzata dei dati provenienti dai sensori del telefono del paziente. Questi dati, trasmessi periodicamente in formato JSON, vengono ricevuti dal server, sottoposti a un processo di validazione per garantirne l'integrità e la coerenza, e successivamente memorizzati in modo sicuro all'interno del database per essere utilizzati nelle analisi successive.

6.2.3 Calcolo delle metriche

Una volta al giorno, a mezzanotte, il server esegue un processo automatico che recupera tutti i dati memorizzati per ogni paziente e calcola una serie di metriche su base oraria per ogni paziente. Questi calcoli vengono effettuati tramite script Python eseguiti come task pianificati, utilizzando la libreria `APScheduler` per garantirne un'esecuzione regolare. Le metriche, in linea con i requisiti di elaborazione dei dati, includono:

- Esposizione alla luce.
- Ampiezza audio media.
- Durata delle conversazioni.
- Ampiezza media dell'accelerometro.
- Utilizzo totale dello schermo.
- Numero di utilizzi del dispositivo.
- Distanza percorsa.
- Numero di luoghi visitati.

Il server, dunque, elabora queste informazioni e le aggrega per ciascun paziente, registrando i risultati nel database per future visualizzazioni. Per garantire una maggiore accuratezza dei dati GPS, è stata utilizzata la tecnica di clusterizzazione DBSCAN. Questo metodo consente di filtrare i punti di rilevamento rumorosi e isolare i movimenti effettivi, assicurando che solo le distanze significative vengano considerate nel calcolo dell'attività fisica.

6.2.4 Dashboard per il medico

Per consentire ai medici di accedere alle metriche elaborate, è stata sviluppata una dashboard web integrata nel server. L'accesso alla dashboard è protetto tramite un sistema di autenticazione basato su username e password,

garantendo che solo i medici autorizzati possano accedere ai dati sensibili dei pazienti. La dashboard offre le seguenti funzionalità:

- **Visualizzazione dei pazienti a rischio**: Dopo l'accesso alla dashboard, il medico può immediatamente visualizzare i pazienti con un rischio elevato di ricaduta. Cliccando su uno dei pazienti, il medico viene reindirizzato a una pagina dedicata alla visualizzazione dettagliata dei dati del paziente.

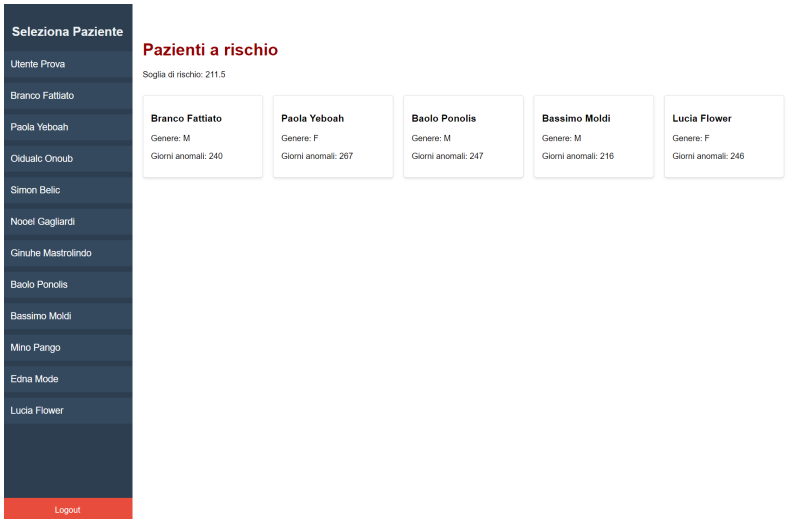

Figura 6.4: Pagina iniziale della dashboard per il medico

- **Selezione del paziente**: Tramite una barra laterale posta sulla sinistra della pagina, il medico può selezionare il paziente di interesse e accedere ai suoi dati.
- **Cambio di data**: È possibile visualizzare i dati relativi a una data specifica tramite un calendario interattivo presente sulla pagina dei dati del paziente.
- **Visualizzazione dei grafici**: Il medico può visualizzare grafici che rappresentano le metriche orarie dei pazienti, così da avere un feedback visivo sull'andamento di un paziente.

- **Caricamento dei file JSON e MP4**: La dashboard consente anche l'upload manuale di file JSON e MP4 che contengono dati di backup o registrazioni audio salvate localmente sul dispositivo del paziente.

Figura 6.5: Pagina contenente le informazioni su un paziente

6.3 Modello di predizione

In questa sezione verrà descritta l'implementazione del modello di predizione. Ogni parte del codice, suddiviso nei vari file, corrisponde alle sezioni teoriche descritte nel Capitolo 5.

6.3.1 Struttura del codice

Il codice del modello di predizione è suddiviso come segue:

- `data_processing.py`: Si occupa del preprocessing dei dati.
- `model.py`: Definisce l'architettura del modello.
- `evaluation.py`: Valuta le prestazioni del modello.
- `utils.py`: Contiene funzioni di utilità.
- `main.py`: Il file principale che gestisce il flusso dell'applicazione.

6.3.2 `data_processing.py`

Il file `data_processing.py` contiene le funzioni necessarie per la preparazione dei dati descritte nel Capitolo 5. Ricapitolando, le operazioni effettuate per la preparazione dei dati sono:

- Riempimento delle righe contenenti valori `NaN`.
- Rimozione dei pazienti con meno di 30 giorni di dati.
- Estrazione delle feature dalla data.

La pipeline di preprocessing si assicura che i dati grezzi vengano trasformati in un formato compatibile con l'input del modello.

6.3.3 `model.py`

In `model.py` viene definita l'architettura del modello LSTM autoencoder. Il file implementa la rete neurale come descritto nel capitolo teorico, con layer di codifica e decodifica LSTM, e include la funzione di ricostruzione che viene utilizzata per identificare le anomalie. È qui che si definiscono i vari parametri del modello, come il numero di unità nascoste, le funzioni di attivazione e le dimensioni degli input.

6.3.4 `evaluation.py`

In `evaluation.py` viene gestita la valutazione delle performance del modello, calcolando le seguenti metriche chiave:

- **Support**: Numero di campioni analizzati.
- **Anomalous Days**: Conteggio dei giorni identificati come anomali dal modello.
- **Anomaly Percentage**: Percentuale di giorni anomali rispetto al totale.
- **Mean Square Error (MSE)**: Media dei quadrati delle differenze tra i dati originali e quelli ricostruiti.

- **Cumulative Error**: Somma cumulativa degli errori di ricostruzione lungo tutto il periodo.
- **Mean Reconstruction Error**: Media degli errori di ricostruzione giornalieri.
- **Std Reconstruction Error**: Deviazione standard degli errori di ricostruzione, che misura la variabilità degli errori rispetto alla media.
- **Average Training Loss**: Media delle perdite (loss) registrate durante la fase di addestramento, che indica la convergenza del modello.

6.3.5 `utils.py`

Il file `utils.py` contiene funzioni di utilità generali che supportano il funzionamento del sistema. In questo file troviamo le funzioni per gestire le seguenti funzionalità:

- Salvare i risultati del modello in un file markdown.
- Salvare i risultati del modello nel database.
- Impostare la label di predizione ai pazienti.

6.3.6 `main.py`

`main.py` rappresenta il punto di ingresso dell'applicazione. Qui viene orchestrato il flusso principale delle operazioni, utilizzando le funzionalità descritte nei file precedenti. Il flusso di eventi è il seguente:

1. Recupero dei dati dal file di input.
2. Preprocessing dei dati per la preparazione all'addestramento.
3. Avvio della validazione LOPO. Per ogni paziente si procede con:
    3.1 Addestramento del modello sui dati di tutti i pazienti eccetto il paziente corrente.

3.2 Predizione utilizzando i dati del paziente escluso dall'addestramento.

3.3 Calcolo dei giorni anomali basato sui dati ricostruiti dal modello.

3.4 Calcolo delle metriche di valutazione per il modello.

3.5 Salvataggio dei giorni anomali rilevati nel totale complessivo.

4. Calcolo della soglia di rischio di ricaduta basata sul numero totale di giorni anomali.

5. Salvataggio dei dati finali nel database.

6.4 Database

In questa sezione, esamineremo il processo di salvataggio dei dati nel database, evidenziando come le classi definite nella sezione 4.4 siano state mappate in modo efficace nelle varie collezioni del database.

6.4.1 Collezione `patient-data`

Descrizione

Una collezione contentente i dati relativi ai pazienti monitorati dal sistema. Ogni documento rappresenta le informazioni relative ad un paziente.

Attributi

- **userId**: Una stringa che rappresenta l' identificatore univoco per il paziente.
- **firstName**: Una stringa che rappresenta il nome del paziente.
- **lastName**: Una stringa che rappresenta il cognome del paziente.
- **gender**: Una stringa che rappresenta il sesso biologico del paziente.

- **birthdate**: Un oggetto `Date` che rappresenta la data di nascita del paziente.
- **address**: Una stringa che rappresenta l'indirizzo di residenza del paziente.
- **EurekaID**: ID numerico assegnato al paziente per il modello di predizione.
- **contacts**: Un array di oggetti contenente le informazioni di contatto relative al paziente:
  - **phoneNumber**: Una stringa che rappresenta il numero di telefono del contatto.
  - **firstName**: Una stringa che rappresenta il nome del contatto.
  - **lastName**: Una stringa che rappresenta il cognome del contatto.
  - **familyMember**: Una stringa che rappresenta il grado di parentela del contatto relativo al paziente. Assume valore "x" nel caso in cui il numero sia del paziente stesso.


```
{
  "_id": {
    "$oid": "66e04c8777529f33fb64d7a9"
  },
  "userId": "824e785f-04ed-4d47-85fa-d29bef77d678",
  "firstName": "Utente",
  "lastName": "Prova",
  "gender": "M",
  "birthdate": {
    "$date": "1997-11-17T00:00:00.000Z"
  },
  "address": "Via Paperon de Paperoni 104 Paperopoli(PP)",
  "contacts": [
    {
      "phoneNumber": "+393564895124",
      "familyMember": "x"
```

```
    },
    {
      "phoneNumber": "+393451226982",
      "firstName": "Utento",
      "lastName": "Prova",
      "familyMember": "Padre"
    },
    {
      "phoneNumber": "+393563415123",
      "firstName": "Utenta",
      "lastName": "Provola",
      "familyMember": "Madre"
    }
  ],
  "EurekaID": {
    "$numberLong": "4"
  }
}
```

Listing 6.1: Esempio di documento contenente i dati del paziente

6.4.2 Collezione `medic-data`

Descrizione

Una collezione contentente i dati relativi ai medici.

Attributi

- **username**: Una stringa che rappresenta lo username del medico per l'accesso alla piattaforma.
- **password**: Una stringa che rappresenta la password del medico per l'accesso alla piattaforma. La password è crittografata utilizzando l'algoritmo di crittografia PBKDF2 (Password-Based Key Derivation Function 2), con SHA256 come funzione di hash, e include l'aggiunta di un salt per migliorare la sicurezza, rendendo difficile la decifrazione.

- **firstName**: Una stringa che rappresenta il nome del medico.
- **lastName**: Una stringa che rappresenta il cognome del medico.


```
{
  "_id": {
    "$oid": "66f98dbfefbfb8bd9ab5cdc8"
  },
  "username": "morenomorello",
  "password": "scrypt:32768:8:1$04B3UsFLJfEqcbFS$73b5cb4c52dcd264
               0912704c9c7aa8996c687b0e43892b547b46a4a773426ef65
               0964de2a5d73bdbbc1c77f217351ac2ed5dfce11549d0c07f
               680ba27feddb85",
  "firstName": "Moreno",
  "lastName": "Morello"
}
```


Listing 6.2: Esempio di documento contenente i dati del medico

6.4.3 Collezione `raw-data`

Descrizione

Una collezione contenente tutti i dati raccolti dai sensori dei dispositivi mobili.

Attributi

- **sensorStartTime**: Un intero che indica l'istante temporale, espresso in *microseconds since epoch*, in cui è avvenuta la misurazione.
- **data**: Un oggetto contenente i dati rilevati dal sensore. La struttura dell'oggetto dipende dal tipo di dato recuperato dal sensore. Ogni oggetto data contiene il campo "__type" che indica di che tipo di misura si tratta.
- **userId**: Una stringa che indica il paziente a cui appartiene la misura.

```
{
  "_id": {
    "$oid": "66e2bbe8acef636089f79bb0"
  },
  "sensorStartTime": {
    "$numberLong": "1720954431602552"
  },
  "data": {
    "__type": "location",
    "latitude": 40.7434881,
    "longitude": 14.632278,
    "altitude": 95.4000015258789,
    "accuracy": 20,
    "verticalAccuracy": 2.39630389213562,
    "speed": 0.21489305794239044,
    "speedAccuracy": 0,
    "heading": 102.43281555175781,
    "headingAccuracy": 0,
    "time": "2024-07-14T12:53:50.188",
    "isMock": false,
    "elapsedRealtimeNanos": 11795337000000,
    "elapsedRealtimeUncertaintyNanos": 0
  },
  "userId": "824e785f-04ed-4d47-85fa-d29bef77d677"
}
```


Listing 6.3: Esempio di documento con dati grezzi relativi alla posizione del paziente

6.4.4 Collezione `processed-data`

Descrizione

Una collezione che contiene le metriche calcolate a partire dai dati grezzi.

Attributi

- **date**: Un oggetto `Date` che indica la data relativa alla metrica.
- **hour**: Un intero che indica l'ora relativa alla metrica.
- **type**: Una stringa che indica il tipo di metrica calcolata.
- **userId**: Una stringa che indica l'utente relativo alla metrica.
- **value**: Un intero che indica il valore della metrica.


```
{
  "_id": {
    "$oid": "66ebf0bc12a582d54a01c64e"
  },
  "date": {
    "$date": "2024-07-14T13:00:00.000Z"
  },
  "hour": 13,
  "type": "distance",
  "userId": "824e785f-04ed-4d47-85fa-d29bef77d677",
  "value": 2.7946832064150384
}
```


Listing 6.4: Esempio di documento con dati elaborati relativi alla distanza percorsa in un'ora

6.4.5 Collezione `prediction-data`

Descrizione

Una collezione che contiene i dati relativi alle predizioni effettuate dal modello di predizione.

Attributi

- **EurekaID**: ID numerico assegnato al paziente per la predizione.

- **Support**: Un intero che indica il numero di campioni analizzati dal modello di predizione.
- **Anomalous Days**: Un intero che indica il numero di giorni anomali rilevati dal modello di predizione.
- **MSE**: Un valore `Double` che indica il Mean Square Error.
- **Anomaly Percentage**: Un valore `Double` che indica la percentuale dei giorni anomali rispetto al totale dei giorni.
- **Cumulative Error**: Un valore `Double` che indica l'errore di ricostruzione cumulativo.
- **Mean Reconstruction Error**: Un valore `Double` che indica la media degli errori di ricostruzione.
- **Std Reconstruction Error**: Un valore `Double` che indica la deviazione standard degli errori di ricostruzione.
- **Average Training Loss**: Un valore `Double` che indica la perdita media durante l'addestramento.
- **Threshold**: Un valore `Double` che indica la soglia di anomalie impostata per la ricaduta.
- **Relapse**: Un valore intero che indica se il paziente è a rischio ricaduta. Questo campo assume i valori 0 o 1, dove 0 indica che il paziente non è a rischio di ricaduta, mentre 1 indica che il rischio di ricaduta è stato rilevato

```
{
  "_id": {
    "$oid": "6709383bec62abf6ff8b5ce6"
  },
  "EurekaID": 5,
  "Support": 467,
  "Anomalous Days": 240,
  "MSE": 0.005311889521164204,
  "Anomaly Percentage": 51.39186295503212,
  "Cumulative Error": 358.01493392915967,
  "Mean Reconstruction Error": 0.047914204219641335,
  "Std Reconstruction Error": 0.039787730171013135,
  "Average Training Loss": 0.010965875536203384,
  "Threshold": 211.5,
  "Relapse": 1
}
```


Listing 6.5: Esempio di documento con dati di predizione

CAPITOLO 7

CONCLUSIONI E SVILUPPI FUTURI

In questo capitolo verranno riepilogati i principali risultati del lavoro svolto, mettendo in luce come il sistema progettato risponda agli obiettivi prefissati e le sue potenzialità. Saranno inoltre proposti suggerimenti per sviluppi futuri volti a migliorare ulteriormente la qualità e l'efficacia del sistema.

7.1 Conclusioni

Il sistema sviluppato soddisfa gli obiettivi delineati nel Capitolo 4, rappresentando una soluzione efficace per il monitoraggio continuo dei pazienti e semplificando l'intervento medico tramite la dashboard dedicata. L'uso dell'imputazione statistica per la predizione consente un intervento proattivo, anche in assenza di dati relativi alle ricadute. Tuttavia, esistono alcune limitazioni che ne riducono l'usabilità. La gestione aggressiva dei processi in background da parte del sistema operativo Android può causare terminazioni impreviste, portando alla perdita di dati. Sebbene il sistema gestisca molti di questi casi, non può garantire la completa eliminazione del problema. Inoltre,

le applicazioni che monitorano continuamente i pazienti infrangono le linee guida del Play Store di Google, richiedendo installazioni manuali e rallentando la distribuzione. Un altro limite riguarda la raccolta dei dati tramite smartphone: i pazienti schizofrenici, in particolare, potrebbero essere riluttanti a partecipare allo studio per timore di essere osservati o potrebbero dimenticare di utilizzare il telefono correttamente, compromettendo l'affidabilità dei dati.

7.2 Sviluppi futuri

Un primo passo per migliorare il sistema sarebbe testarlo su pazienti reali, raccogliendo dati effettivi tramite uno studio di 12 mesi, in maniera simile a CrossCheck [5], per ottenere una quantità sufficiente di dati dai sensori, oltre che informazioni sulle ricadute. Con dati sulle ricadute, si potrebbe addestrare un modello più sofisticato, utilizzando tecniche come i Transformer, affiancandolo a quello proposto nel Capitolo 5. L'integrazione di dispositivi aggiuntivi, come SmartWatch o i più recenti Smart Rings, aumenterebbe l'affidabilità dei dati, ma è poco realistico per via dei costi e della natura dei pazienti schizofrenici, che potrebbero essere riluttanti nell'indossarli. Infine, sarebbe utile includere sondaggi periodici per i pazienti, offrendo ai medici uno strumento simile alle interviste cliniche utilizzate con pazienti schizofrenici. Attraverso questi sondaggi, si potrebbe prevedere in modo più accurato lo score BPRS. Questa integrazione fornirebbe ai medici ulteriori dati soggettivi sulle condizioni mentali dei pazienti, permettendo una valutazione più completa, migliorando così la capacità di identificare segnali di allarme per una possibile ricaduta. Tuttavia, anche questa integrazione presenta delle sfide: l'adesione da parte dei pazienti potrebbe essere incostante, e si potrebbe incorrere nelle limitazioni di Android per quanto riguarda la gestione in background dell'applicazione.

BIBLIOGRAFIA

[1] World Health Organization. *Schizophrenia*. 2023. URL: https://www.who.int/news-room/fact-sheets/detail/schizophrenia.

[2] Glynn Harrison et al. «Recovery from psychotic illness: a 15- and 25-year international follow-up study». In: *The British Journal of Psychiatry* 178 (2001), pp. 506–517. DOI: 10.1192/bjp.178.6.506.

[3] John E Overall e Donald R Gorham. «The Brief Psychiatric Rating Scale (BPRS): Recent developments in ascertainment and scaling». In: *Psychopharmacology Bulletin* 24.1 (1988), pp. 97–99.

[4] Souhaiel Bouhlel et al. «Les prodromes des rechutes schizophréniques: étude descriptive et comparative [Prodromal symptoms in schizophrenic relapse: A descriptive and comparative study]». In: *L'Encephale* 38.5 (2012), pp. 397–403. DOI: 10.1016/j.encep.2011.12.005. URL: https://doi.org/10.1016/j.encep.2011.12.005.

[5] Rui Wang et al. «Predicting Symptom Trajectories of Schizophrenia using Mobile Sensing». In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1.3 (set. 2017). DOI: 10.1145/3130976. URL: https://doi.org/10.1145/3130976.

[6] John Torous et al. «New Tools for New Research in Psychiatry: A Scalable and Customizable Platform to Empower Data Driven

Smartphone Research». In: *JMIR Mental Health* 3.2 (mag. 2016), e16. ISSN: 2368-7959. DOI: 10.2196/mental.5165. URL: http://www.ncbi.nlm.nih.gov/pubmed/27150677.

[7] Anna M. Langener et al. «A template and tutorial for preregistering studies using passive smartphone measures». In: *Behavior Research Methods* (ago. 2024). ISSN: 1554-3528. DOI: 10.3758/s13428-024-02474-5. URL: https://doi.org/10.3758/s13428-024-02474-5.

[8] MIT Media Lab. *Funf Open Sensing Framework Blog*. Retrieved: 2024-10-08. 2011.

[9] Onnela lab. *Digital Phenotyping and Beiwe Research Platform*. Retrieved: 2024-10-13. 2011.

[10] Nicholas Lane et al. «BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing». In: IEEE, apr. 2012. DOI: 10.4108/icst.pervasivehealth.2011.246161.

[11] Bibek Lamichhane, Jiawei Zhou e Akane Sano. «Psychotic Relapse Prediction in Schizophrenia Patients Using A Personalized Mobile Sensing-Based Supervised Deep Learning Model». In: *IEEE Journal of Biomedical and Health Informatics* 27.7 (2023), pp. 3246–3257. DOI: 10.1109/JBHI.2023.3265684. URL: https://doi.org/10.1109/JBHI.2023.3265684.

[12] Joanne Zhou et al. «Predicting Psychotic Relapse in Schizophrenia With Mobile Sensor Data: Routine Cluster Analysis». en. In: *JMIR Mhealth Uhealth* 10.4 (apr. 2022), e31006.

[13] Bishal Lamichhane et al. «Patient-Independent Schizophrenia Relapse Prediction Using Mobile Sensor Based Daily Behavioral Rhythm Changes». In: *Wireless Mobile Communication and Healthcare*. A cura di Juan Ye et al. Cham: Springer International Publishing, 2021, pp. 18–33. ISBN: 978-3-030-70569-5.

[14] Jakob E. Bardram. ≪The CARP Mobile Sensing Framework – A Cross-platform, Reactive, Programming Framework and Runtime Environment for Digital Phenotyping≫. In: (2020). arXiv: 2006.11904 [cs.HC]. URL: https://arxiv.org/abs/2006.11904.

[15] C. S. Team. *Crosscheck Kaggle Dataset*. 2020. URL: https://www.kaggle.com/datasets/dartweichen/crosscheck.

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE