iWalk

Repository GitHub

Roberto Ambrosino
Matricola: 0512117886

Corso di
Fondamenti di
Intelligenza Artificiale
2024/2025

Indice

1 Introduzione

Camminare è una delle attività fisiche più semplici e accessibili, con benefici significativi per la salute. Oggi, grazie a dispositivi come gli smartwatch, è possibile raccogliere una grande quantità di dati relativi al movimento e al dispendio calorico, semplicemente indossandoli. Nonostante ciò, solo una parte della popolazione mondiale raggiunge gli obiettivi quotidiani consigliati dall'Organizzazione Mondiale della Sanità.

2 Definizione del problema

2.1 Obiettivi

L'obiettivo è creare un'applicazione per iPhone in grado di comunicare all'utente quanti passi effettuare per bruciare determinate calorie.

Il progetto si concentra sull'implementazione di un modello di regressione lineare utilizzando CreateML, il framework di Apple per il machine learning, sulla base dei dati sanitari raccolti da uno smartwatch.

In aggiunta a ciò, l'utente potrà interagire con un LLM per avere una stima di quante calorie ha assunto ogni giorno, sulla base di ciò che mangia.

2.2 Specifica PEAS

Di seguito è riportata la specifica PEAS dell'applicazione:

- **Performance**: l'accuratezza delle previsioni viene misurata tramite k-fold validation, adottando il Mean Absolute Error (MAE) per fornire all'utente una valutazione intuitiva
- **Environment**: gli elementi che costituiscono l'ambiente sono i dati raccolti da uno smartwatch
- **Actuators**: generazione di una previsione del numero di passi da effettuare sulla base delle calorie da bruciare
- **Sensors**: smartwatch dotato di sensori biometrici e di movimento

2.3 Descrizione dell'ambiente

Di seguito è descritto l'ambiente in cui opera l'applicazione:

- **Parzialmente osservabile**: l'applicazione non ha accesso a tutte le variabili che influenzano il numero di passi, come il tipo di attività svolta o il terreno, ma solo alle calorie bruciate e ai minuti di esercizio giornalieri.
- **Stocastico**: l'applicazione non è a conoscenza di cosa succede all'ambiente nel momento in cui viene effettuata una previsione
- **Episodico**: ogni previsione è indipendente dalle altre
- **Statico**: i dati di input non cambiano mentre il modello effettua la previsione
- **Continuo**: i dati di input sono valori numerici su un intervallo continuo
- **Mono-agente**: il modello di Machine Learning è l'unico agente coinvolto responsabile della previsione

2.4 Analisi del problema

L'obiettivo dell'applicazione è prevedere il numero di passi a partire dalle calorie da bruciare, utilizzando un approccio basato sul machine learning. Questo problema è un'istanza dei problemi di apprendimento supervisionato, più precisamente nella regressione, in quanto l'output è un valore numerico continuo.

Per quanto concerne la scelta di un modello, si è scelto di adottare un modello di regressione lineare multipla, in quanto vengono usati i dati relativi alle calorie bruciate e ai minuti di esercizio giornalieri per prevedere i passi da fare.

In aggiunta a ciò, per aiutare l'utente nel tenere traccia delle calorie assunte giornalmente in modo naturale, si è scelto di implementare nell'applicazione il supporto ad un LLM.

Il principale ostacolo riscontrato nella realizzazione dell'applicazione, è stato determinato dalle modalità di creazione del dataset, in quanto non sempre tutti i dati memorizzati dall'iPhone si sono dimostrati completi.

3 Analisi dei Dati

Prima di procedere con l'addestramento del modello, è stata effettuata un'analisi esplorativa del dataset, al fine di comprendere meglio la distribuzione delle variabili e le relazioni tra le feature.

3.1 Scelta delle feature

È importante sottolineare che le metodologie con cui gli smartwatch raccolgono e calcolano i dati non sono di pubblico dominio. Ogni produttore adotta algoritmi proprietari che combinano dati di sensori biometrici con parametri personali come età, peso, altezza o sesso per stimare i valori mostrati all'utente. Tuttavia, non essendo accessibili nel dettaglio questi algoritmi, risulta difficile costruire un modello preciso e generalizzabile basato su dati interni e specifici di un solo dispositivo. Per questo motivo, ai fini dell'applicazione, è stato scelto di utilizzare come feature solo tre variabili comuni nella maggior parte degli smartwatch: le calorie bruciate, i minuti di esercizio e i passi. Questa scelta consente una maggiore compatibilità e adattabilità dell'app a diversi modelli e marchi di smartwatch.

3.2 Premessa sull'origine dei dati

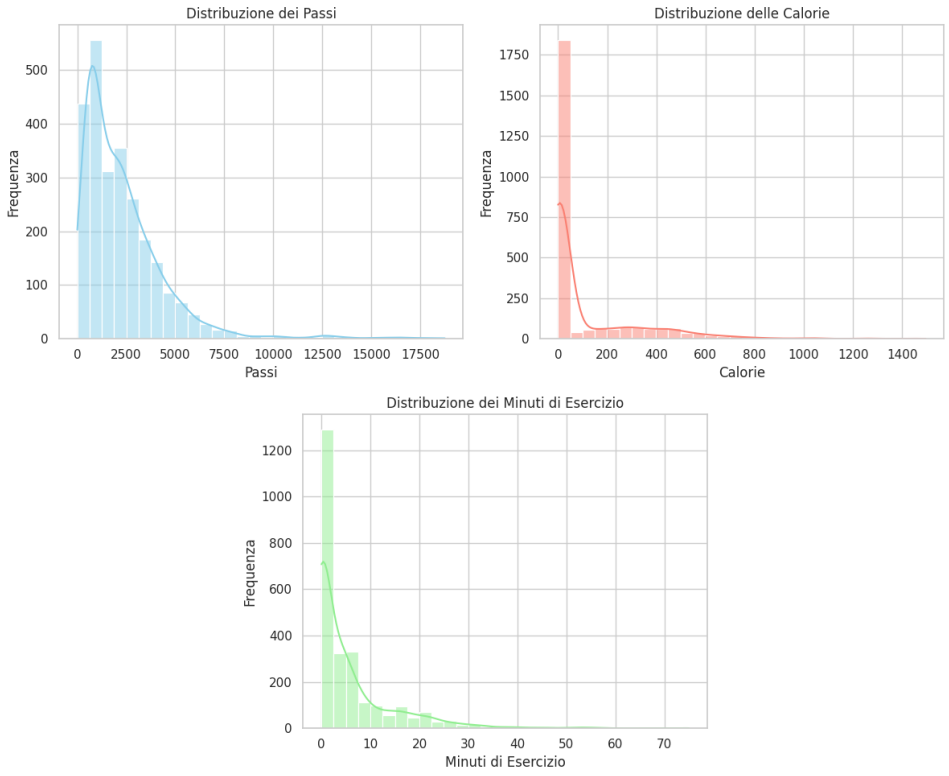
I dati utilizzati per l'addestramento e l'analisi del modello provengono da un dataset personale, raccolto attraverso l'utilizzo quotidiano di uno smartwatch. È importante sottolineare che il soggetto in questione presenta uno stile di vita prevalentemente sedentario, con livelli di attività fisica piuttosto ridotti. Inoltre, l'utilizzo dello smartwatch non è stato costante: in diverse giornate il dispositivo non è stato indossato, oppure è stato utilizzato in modo discontinuo, comportando la registrazione parziale o nulla delle attività quotidiane. Questi fattori hanno influito sulla qualità e completezza del dataset originale, giustificando le successive operazioni di pulizia, imputazione e validazione dei dati, necessarie per garantire risultati più affidabili nel processo di modellazione.

È importante sottolineare che l'applicazione non si basa su un modello preaddestrato su un dataset generico. Al contrario, ogni utente genera dinamicamente il proprio dataset personale attraverso i dati raccolti quotidianamente dallo smartwatch. Questo approccio consente di creare un modello su misura, ottimizzato per il profilo fisico e le abitudini dell'utente stesso.

Di conseguenza, il dataset presentato in questo report non rappresenta una limitazione del sistema in generale, ma un esempio di funzionamento su un caso reale. L'applicazione è stata progettata per adattarsi progressivamente al singolo individuo.

3.3 Analisi del dataset grezzo

Il dataset iniziale, ottenuto direttamente dai dati raccolti dallo smartwatch, presenta criticità che ne compromettono l'utilizzo diretto per l'addestramento di un modello affidabile. In particolare, si osserva una forte presenza di valori molto bassi o prossimi allo zero per tutte le variabili, spesso dovuti a giornate in cui l'utente è rimasto sedentario o non ha indossato il dispositivo. Per visualizzare meglio questa situazione, sono stati generati grafici che evidenziano tale squilibrio e la necessità di un intervento di pulizia e selezione dei dati.



La distribuzione dei dati risulta quindi sbilanciata, con una concentrazione eccessiva di campioni con attività fisica scarsa o nulla, il che può influenzare negativamente le prestazioni del modello.

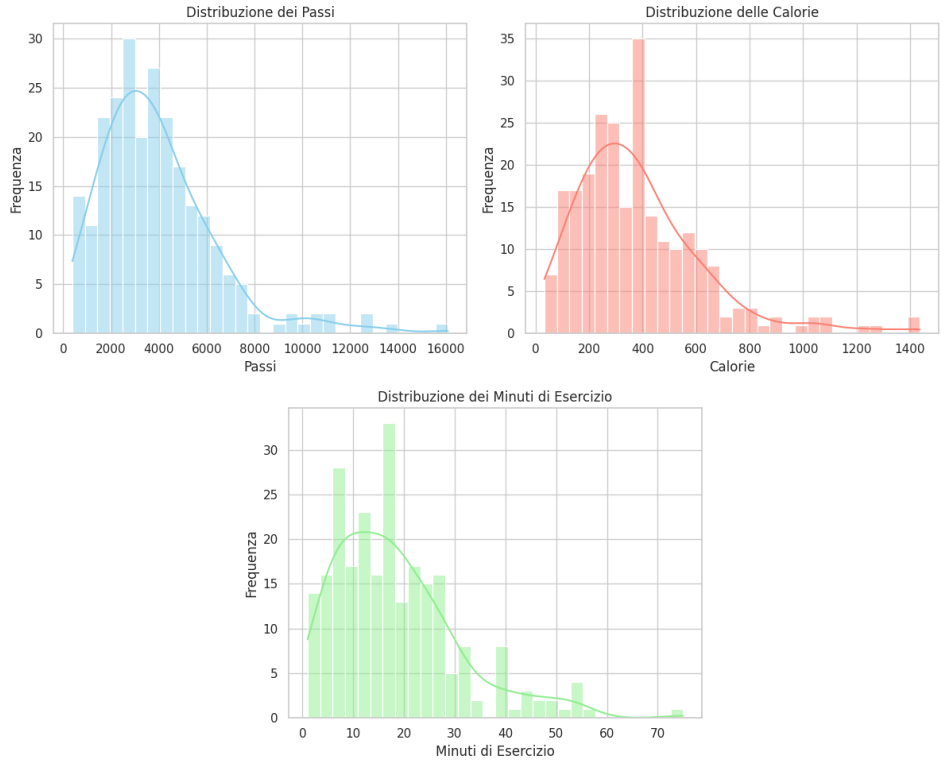
Prima della pulizia dei dati, sono state effettuate ulteriori analisi per evidenziare ulteriormente le criticità del dataset grezzo.

	Passi	Minuti di Esercizio	Calorie
count	2558.00	2558.00	2558.00
mean	2335.71	5.64	111.35
std	2305.39	9.76	265.73
min	0.00	0.00	0.00
25%	765.25	0.00	0.00
50%	1820.50	2.00	11.00
75%	3113.50	7.00	136.00
max	18758.00	75.00	1495.00

In aggiunta alle criticità precedentemente descritte, si osserva la presenza di giornate attive alternate a molte altre pressoché sedentarie o in cui lo smartwatch non è stato indossato. Sebbene la variabilità nei comportamenti giornalieri sia un aspetto positivo per l'addestramento del modello, la presenza di numerosi record con valori prossimi allo zero rischia di influenzare negativamente le capacità predittive. Per questo motivo, si è ritenuto opportuno intervenire con un processo di pulizia, rimuovendo i record anomali o poco significativi, al fine di ottenere un dataset migliore per l'addestramento del modello.

3.4 Analisi del dataset pulito

A seguito della fase di pulizia e selezione, è stato ottenuto un dataset che rappresenta in modo più realistico l'attività fisica quotidiana di un utente. I valori estremamente bassi o nulli, potenzialmente dovuti a giornate di inattività o a dati mancanti, sono stati rimossi.



	Passi	Minuti di Esercizio	Calorie
count	246.00	246.00	246.00
mean	3926.43	18.33	383.16
std	1954.88	10.42	209.11
min	377.00	1.00	33.00
25%	2339.50	9.00	224.25
50%	3588.50	16.00	340.50
75%	5030.50	24.00	493.75
max	16085.00	75.00	1437.00

Le statistiche evidenziano un dataset più coerente:

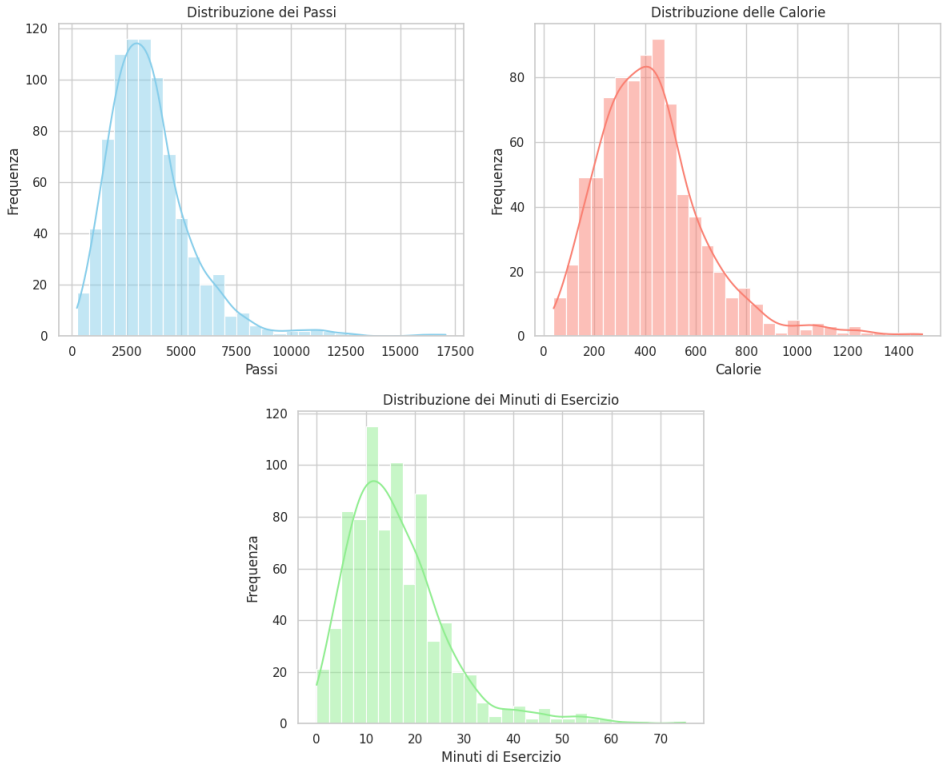
- La media dei passi giornalieri (3926) e delle calorie bruciate (383) riflette una moderata attività fisica coerente con la tipologia di utente dal quale è derivato il dataset.
- La deviazione standard indica una variabilità accettabile tra giornate più o meno attive.
- I valori minimi sono tutti superiori a zero, a conferma dell'avvenuta rimozione dei casi non significativi.
- L'intervallo interquartile (tra il 25° e il 75° percentile) mostra che la maggior parte dei giorni presenta valori coerenti e realistici, evitando la distorsione causata da outlier.

Questo nuovo dataset rappresenta una base più affidabile per l'addestramento del modello predittivo, migliorando sia l'accuratezza che la generalizzabilità dell'algoritmo.

Dopo aver rimosso i record ritenuti anomali o non significativi , si è ottenuto un dataset composto da 246 record. I valori risultano più coerenti e rappresentativi di giornate in cui l'utente ha effettivamente svolto attività fisica, tuttavia, la quantità ridotta di dati potrebbe limitare la capacità del modello di apprendere schemi generali e adattarsi a differenti livelli di attività. Questa limitazione ha motivato l'introduzione di una procedura di imputazione dei dati tramite modello, con l'obiettivo di aumentare la dimensione del dataset mantenendo coerenza e realismo nei nuovi record generati.

3.5 Analisi del dataset con imputazione tramite modello

Applicando una tecnica di imputazione basata su modello, è stato possibile ricostruire o simulare alcuni record mancanti, portando il dataset a un totale di 809 osservazioni.



Le distribuzioni delle tre feature nel dataset mostrano un comportamento coerente con i dati sull'attività fisica quotidiana. La maggior parte dei valori si concentra su livelli moderati di attività (es. 2000–5000 passi, 10–25 minuti di esercizio), mentre i valori estremi verso destra rappresentano giornate più intense ma meno frequenti. La forma delle distribuzioni, priva di irregolarità evidenti, conferma la buona qualità del dataset dopo l'imputazione.

	Passi	Minuti di Esercizio	Calorie
count	809.00	809.00	809.00
mean	3546.67	16.14	423.12
std	1954.42	10.10	210.31
min	227.00	0.00	39.00
25%	2274.00	9.00	279.00
50%	3231.00	14.00	400.00
75%	4275.00	21.00	518.00
max	17077.00	75.00	1495.00

Le statistiche descrittive confermano un buon bilanciamento: i valori medi risultano compatibili con una normale attività fisica giornaliera di un utente tendenzialmente sedentario, mentre la deviazione standard resta contenuta, indicando una variabilità realistica ma priva di eccessi. I valori minimi e massimi sono plausibili e coerenti con l'attività di una persona poco attiva. Questo dataset rappresenta un buon compromesso tra quantità e qualità, adatto per l'addestramento di un modello più robusto.

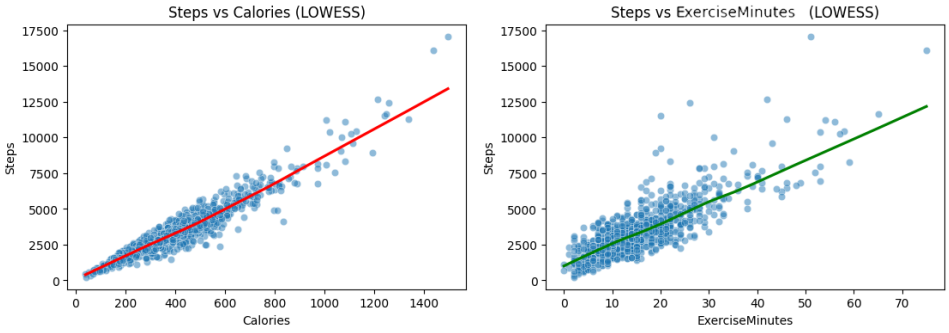
3.6 Relazioni tra le feature

Per studiare le relazioni tra le feature passi, minuti di esercizio e calorie bruciate, è stata calcolata la matrice delle correlazioni.

	Passi	**Calorie**	**Minuti di Esercizio**
Passi	-	0.9479	0.8145
Calorie	0.9479	-	0.7392
Minuti di Esercizio	0.8145	0.7392	-

I risultati mostrano una forte correlazione positiva tra passi e calorie ($r = 0{,}9479$), e una correlazione leggermente inferiore ma comunque significativa tra passi e minuti ($r = 0{,}8145$). Questi risultati indicano che entrambe le variabili indipendenti sono buoni predittori del numero di passi, con le calorie che forniscono l'informazione più forte.

A supporto della matrice di correlazione, sono stati realizzati anche scatter plot con sovrapposta una curva di tendenza LOWESS, che evidenziano la natura tendenzialmente lineare di entrambe le relazioni.



Nel corso dell'analisi preliminare sono state considerate anche altre variabili potenzialmente rilevanti, come il numero di piani saliti o la frequenza cardiaca media (bpm), al fine di valutare un loro eventuale contributo al miglioramento della capacità predittiva del modello. Tuttavia, un esame delle correlazioni non ha evidenziato una relazione statisticamente significativa tra tali feature e il numero di passi. Di conseguenza, si è optato per escluderle dal modello finale.

4 Soluzione del problema

Per permettere all'applicazione di poter apprendere sulla base dei dati dei singoli utenti, è stato necessario innanzitutto definire le modalità di creazione del dataset. In particolare, si è reso necessario effettuare operazioni di Feature Engineering come rimozione degli outlier e generazione di dati sintetici.

Successivamente sono state implementate funzionalità come la possibilità di interagire con un LLM per ottenere le calorie assunte giornalmente mediante linguaggio naturale, possibilità di testare il modello con valori specificati dall'utente e un'interfaccia grafica intuitiva per il funzionamento generale dell'applicazione.

4.1 Tecnologie utilizzate

Si è scelto di realizzare l'applicazione interamente in Swift, per poter sfruttare al meglio l'integrazione nativa con il sistema operativo iOS.

L'interfaccia grafica è realizzata mediante il framework SwiftUI.

Per quanto riguarda il framework adottato per il machine learning, si è scelto di utilizzare CoreML, un framework di Apple che permette di integrare modelli di machine learning direttamente nelle applicazioni, consentendo di eseguire previsioni su dispositivi Apple con prestazioni ottimizzate.

L'utente può valutare la precisione delle previsioni del modello grazie al Mean Absolute Error (MAE) calcolato tramite k-fold validation. Questo valore fornisce una misura dell'accuratezza del modello, indicando in media di quanto le previsioni si discostano dai valori reali.

Il LLM scelto per aiutare l'utente nella stima delle calorie assunte giornalmente, è GPT di OpenAI, in particolare nella versione 3.5 Turbo, il modello più economico offerto, per ridurre al minimo i costi in fase di sviluppo.

4.2 Scelta del modello di regressione

Create ML, il framework di machine learning sviluppato da Apple e integrato nell'ecosistema Swift, mette a disposizione una selezione limitata ma ben ottimizzata di modelli per la regressione. In particolare, le uniche opzioni attualmente offerte sono la regressione lineare e il modello basato su Boosted Trees. Nella fase di sviluppo dell'app, sono stati testati entrambi i modelli per confrontarne le prestazioni e valutarne l'idoneità rispetto agli obiettivi del progetto.

4.2.1 Regressione lineare

Per valutare l'efficacia del modello di regressione lineare, sono stati presi in considerazione alcuni indicatori numerici, scelti perché semplici da interpretare e utili per descrivere diversi aspetti dell'errore. Tra questi, il coefficiente di determinazione (R^2) mostra quanto bene il modello riesca a seguire l'andamento reale dei dati. MAE e RMSE, invece, danno un'idea di quanto il modello possa sbagliare, rispettivamente in media e nei casi più estremi.

R^2	**MAE**	**RMSE**
0.9087	297.67	480.75

- Si nota un coefficiente di determinazione R^2 pari a 0.9087: questo significa che circa il 90% della variabilità nei dati è spiegata dal modello, un risultato molto buono considerando la semplicità della regressione lineare e la natura dei dati utilizzati.
- Il MAE, pari a circa 298 passi, indica che in media l'errore tra le previsioni e i valori reali è inferiore a 300 passi, un margine accettabile per un'applicazione rivolta a un uso quotidiano.
- Anche il valore dell'RMSE, poco sopra i 480 passi, conferma che non ci sono grandi deviazioni tra i valori previsti e quelli effettivi. Complessivamente, i numeri ottenuti mostrano che il modello riesce a mantenere una buona precisione e stabilità, risultando adatto allo scopo pratico per cui è stato progettato.

4.2.2 Boosted Trees

Oltre al modello di regressione lineare, è stato sperimentato anche un modello di regressione basato su Boosted Trees, una tecnica che combina più alberi decisionali in sequenza per migliorare progressivamente l'accuratezza predittiva. Sebbene più complesso, questo tipo di modello è supportato nativamente da Create ML e può essere una valida alternativa nei casi in cui la relazione tra le variabili non sia strettamente lineare.

Modello	R^2	**MAE**	**RMSE**
Regressione lineare	0.9087	297.67	480.75
Boosted Trees	0.9221	244.13	390.40

Dal confronto emergono differenze:

- Il modello Boosted Trees mostra un valore di R^2 più alto (0.9221), che indica una maggiore capacità di spiegare la variabilità nei dati rispetto alla regressione lineare.
- Anche gli errori assoluti risultano più contenuti: il MAE scende a circa 244 passi e l'RMSE si attesta attorno a 390, segno che il modello gestisce meglio anche le previsioni meno accurate.

Questi risultati indicano che, pur essendo il modello Boosted Trees più preciso, il miglioramento ottenuto non è tale da giustificare l'aumento della complessità computazionale. I tempi di esecuzione risultano moderatamente più lunghi, rendendo questo approccio meno adatto per un'applicazione pensata per funzionare in modo rapido e reattivo su dispositivi mobili. Considerata la buona affidabilità già offerta dalla regressione lineare, unita alla sua leggerezza e immediatezza, si è scelto di adottare questo modello come soluzione finale per garantire un bilanciamento efficace tra accuratezza e prestazioni.

4.3 Creazione del Dataset

Per l'accesso ai dati sanitari memorizzati sul dispositivo, Apple mette a disposizione il framework HealthKit. Una volta recuperati i dati sanitari dell'utente, aggregati per giorno, questi vengono sottoposti ad una funzione di rimozione degli outlier ed opzionalmente ad una funzione per aumentare il numero di record del dataset.

4.3.1 Feature del Dataset

Il dataset utilizzato per la previsione dei passi include tre feature principali: kcal bruciate, minuti di esercizio e passi, aggregati per giorno. L'obiettivo del modello è di utilizzare le prime due feature, ovvero le kcal bruciate e il tempo trascorso in esercizio, per prevedere il numero di passi da compiere. Ogni record quindi si compone dei seguenti campi:

```
struct HealthData: Codable {
    var steps: Int
    var exerciseMinutes: Int
    var calories: Int
}
```

4.3.2 Rimozione degli outlier

Di seguito è riportata la funzione utilizzata per la rimozione degli outlier:

```
private func removeOutliers(from data: [HealthData], tolerance:
     Double) -> [HealthData] {
    let tempData = data.filter { record in
        record.calories > 30
    }

    var kcalsPerStep = 0.0
    for record in tempData {
        kcalsPerStep += Double(record.calories) / Double(record
    .steps)
    }
    kcalsPerStep /= Double(tempData.count)

    let lowerBound = kcalsPerStep - tolerance

    return tempData.filter { record in
        let kcalsPerStep = Double(record.calories) / Double(
    record.steps)
        return kcalsPerStep >= lowerBound
    }
}
```

Il compito principale di questa funzione è la rimozione dal dataset di record non completi. L'iPhone infatti continua a salvare in memoria alcuni dati relativi al movimento anche quando non è indossato lo smartwatch: vengono infatti registrati i dati relativi ai passi effettuati, ma non quelli relativi alle kcal bruciate. La funzione agisce rimuovendo tutti i record contenenti meno di 30 kcal bruciate, in quanto si sono dimostrati indice di un utilizzo parziale dello smartwatch. Successivamente viene calcolata la media di kcal bruciate per passo e vengono tenuti in considerazione solo i record che rientrano in un determinato intervallo di tolleranza.

4.3.3 Imputazione tramite modello

Di seguito è riportata la funzione utilizzata per la generazione di record sintetici:

```
func modelImputation(from realData: [HealthData], count: Int)
    async throws -> [HealthData] {
    guard !realData.isEmpty else {
        return []
    }

    var syntheticData: [HealthData] = []

    let regressor = try await createTempModel(data: realData)

    for _ in 0..<count {
        // Record casuale
        let randomIndex = Int.random(in: 0..<realData.count)
        let realRecord = realData[randomIndex]

        // Calcola variazione casuale (entro +-15%)
        let variation = Double.random(in: 0.85...1.15)

        // Applica le variazioni
        let syntheticExercise = Int(Double(realRecord.
    exerciseMinutes) * variation)
        let syntheticCalories = Int(Double(realRecord.calories)
     * variation)

        let syntheticRecord = HealthData(
            date: Date(),
            steps: try makePrediction(exerciseMinutes:
    syntheticExercise, calories: syntheticCalories, model:
    regressor.model),
            exerciseMinutes: syntheticExercise,
            calories: syntheticCalories
        )
        syntheticData.append(syntheticRecord)
    }

    return syntheticData
}
```

Per ampliare il dataset e rendere l'addestramento del modello più solido, è stata sviluppata una funzione che genera dati sintetici a partire da quelli reali. La funzione modelImputation prende in input un insieme di record di dati reali sull'attività fisica dell'utente e produce un numero personalizzabile di nuovi record, simulando dati credibili.

Il funzionamento è semplice ma efficace: inizialmente viene costruito un modello temporaneo di regressione lineare, addestrato sui dati reali ottenuti in input. Successivamente, per ogni nuovo record da creare, viene scelto casualmente un record esistente e su di esso si applica una piccola variazione casuale (entro un intervallo del $\pm 15\%$) ai valori di calorie e minuti di esercizio. Questi nuovi valori vengono quindi usati come input del modello, il quale restituisce una stima del numero di passi da effettuare.

Questo approccio permette di ottenere dati nuovi ma coerenti con l'andamento del dataset reale, mantenendo le relazioni statistiche tra le variabili. In questo modo si evita di generare valori slegati dalla realtà, e allo stesso tempo si migliora la capacità del modello di generalizzare su dati leggermente diversi da quelli osservati.

4.4 Creazione del modello di regressione

```
private func createModel() async throws {
    let fileManager = FileManager.default
    let documentsDirectory = try getDocumentsDirectory()
    let csvURL = documentsDirectory.appendingPathComponent("
    HealthData.csv")

    // Verifica che il file esista
    guard fileManager.fileExists(atPath: csvURL.path) else {
        throw CustomError.csvNotFound
    }

    let outputModelURL = documentsDirectory.
    appendingPathComponent("StepsPredictor.mlmodel")
    let compiledModelURL = documentsDirectory.
    appendingPathComponent("StepsPredictor.mlmodelc")

    // Carica il file CSV in un DataFrame
    let dataFrame = try DataFrame(contentsOfCSVFile: csvURL)

    // Crea il modello di regressione
    let outputModel = try MLLinearRegressor(trainingData:
    dataFrame, targetColumn: "Steps")

    // Salva il modello nella cartella Documents
    try outputModel.write(to: outputModelURL)

    // Compila il modello
    let compiledModel = try await MLModel.compileModel(at:
    outputModelURL)

    // Salva il modello compilato nella cartella Documents, se
    ne esiste gia' uno lo elimina
    if fileManager.fileExists(atPath: compiledModelURL.path) {
        try fileManager.removeItem(at: compiledModelURL)
    }
    try fileManager.moveItem(at: compiledModel, to:
    compiledModelURL)
}
```

Il dataset viene salvato in un file csv che verrà poi usato per la creazione del modello. Una volta creato il modello, quest'ultimo viene compilato: CoreML infatti permette di ottimizzare ulteriormente i modelli creati compilandoli sulla base del dispositivo in uso: questo permette di ottimizzare prestazioni, consumi energetici, tempi di caricamento e memoria utilizzata.

4.5 Calcolo del Mean Absolute Error (MAE)

```
func kFoldValidation(data: [HealthData], k: Int) throws -> Int {
    let totalRows = data.count
    let foldSize = totalRows / k
    var errors: [Int] = []

    let shuffledData = data.shuffled()

    for i in 0..<k {
        // Calcolare gli indici per il testSet del fold corrente
        let start = i * foldSize
        let end = min(start + foldSize, totalRows)

        // Creare il test set per il fold corrente
        let testSet = Array(shuffledData[start..<end])

        // Creare il training set rimuovendo le righe del test set
        var trainingData: [HealthData] = []
        trainingData.append(contentsOf: shuffledData[0..<start])
        trainingData.append(contentsOf: shuffledData[end..<totalRows])

        var trainingSet = DataFrame()
        // Aggiungi le colonne una per una, convertendo gli array in formati accettati
        trainingSet.append(column: Column(name: "Steps", contents: trainingData.map { Double($0.steps) }))
        trainingSet.append(column: Column(name: "ExerciseMinutes", contents: trainingData.map { Double($0.exerciseMinutes) }))
        trainingSet.append(column: Column(name: "Calories", contents: trainingData.map { Double($0.calories) }))

        // Addestrare il modello
        let regressor = try MLLinearRegressor(trainingData: trainingSet, targetColumn: "Steps")

        // Calcolo errore medio assoluto (MAE) sul fold
        errors.append(try MAE(data: testSet, model: regressor.model))
    }
    // Calcolare l'errore medio su tutti i fold
    return Int(errors.reduce(0, +) / errors.count)
}
```

```swift
func MAE(data: [HealthData], model: MLModel? = nil) throws ->
    Int {
    var sum = 0

    for record in data {
        let predictedSteps = try makePrediction(
            exerciseMinutes: record.exerciseMinutes,
            calories: record.calories,
            model: model
        )
        let absoluteError = abs(predictedSteps - record.steps)
        sum += absoluteError
    }

    return Int(sum / data.count)
}
```

Prima di calcolare l'errore medio assoluto (MAE), i dati vengono mescolati in modo casuale. Questo passaggio è importante per evitare che l'eventuale ordine dei dati nel dataset influenzi le valutazioni delle performance del modello.

Per la validazione, è stato adottato il metodo della k-fold cross-validation, che suddivide il dataset in più sottoinsiemi (fold) e valuta il modello su ciascuno di essi a rotazione. Questo approccio è stato scelto perché consente di ottenere una stima più affidabile delle prestazioni complessive, evitando che la valutazione dipenda troppo di una particolare suddivisione dei dati tra training e test set. Inoltre, permette di sfruttare l'intero dataset sia per l'addestramento sia per la validazione, riducendo al minimo lo spreco di informazioni.

Per misurare la precisione del modello generato, è stato utilizzato il MAE, in quanto rappresenta un errore medio facilmente interpretabile: indica direttamente di quanti passi, in media, le previsioni si discostano dai valori reali. Il MAE restituisce un'informazione più immediata e comprensibile anche da parte di utenti non esperti, facilitando l'analisi dell'affidabilità del modello in un contesto pratico.

4.6 LLM

4.6.1 Scelta del modello

Per stimare le calorie assunte in base a una descrizione testuale dei pasti, è stata implementata un'integrazione con il LLM GPT-3.5 Turbo, fornito da OpenAI. Questa scelta nasce dalla necessità di interpretare input scritti in linguaggio naturale, spesso informale o poco strutturato, come "ho mangiato una pizza margherita con prosciutto" o "una ciotola di cereali con latte". GPT-3.5 si è dimostrato adatto a questo tipo di compito grazie alla sua capacità di comprendere il contesto e restituire una stima calorica coerente, anche in presenza di varianti negli ingredienti o nelle porzioni. Oltre alla buona accuratezza, ha il vantaggio di essere veloce nella risposta e meno costoso rispetto a modelli più avanzati, il che lo rende più pratico per un'app mobile. In sintesi, la scelta è stata guidata da un equilibrio tra affidabilità, efficienza e semplicità di integrazione nel progetto.

4.6.2 Formulazione del prompt

Un aspetto particolarmente rilevante nello sviluppo dell'integrazione con un LLM, è stato la formulazione del prompt da inviare al modello. Poiché il costo delle API dipende direttamente dal numero di token utilizzati, è stato necessario trovare un giusto equilibrio tra chiarezza della richiesta e sintesi dell'input. Prompt troppo generici portavano a risposte verbose o non focalizzate, mentre prompt troppo stringati potevano ridurre la qualità o la precisione della stima calorica. L'ottimizzazione è quindi passata attraverso vari tentativi, con l'obiettivo di minimizzare il numero di token mantenendo però l'efficacia informativa. Questo ha permesso non solo di ottenere risposte più pertinenti e concise, ma anche di contenere i costi di utilizzo dell'API, rendendo il sistema più sostenibile dal punto di vista economico, soprattutto in un'app destinata a un uso continuativo.

4.6.3 Confronto tra prompt

Prompt	Risposte con testo indesiderato (su 20)	Costo stimato per chiamata (USD)
Prompt 1	0/20	$0.00010
Prompt 2	6/20	$0.00009
Prompt 3	5/20	$0.00012
Prompt 4	4/20	$0.00011

- **Prompt 1 (usato nell'app)**: Rispondi solo con il numero intero totale delle calorie degli alimenti forniti, senza aggiungere mai del testo di nessun tipo. Se non ci sono alimenti validi o sei incerto rispondi con 0.
- **Prompt 2**: Fornisci soltanto il numero totale delle calorie degli alimenti elencati. Non scrivere altro testo. Se l'informazione non è chiara, rispondi con 0.
- **Prompt 3**: Calcola le calorie totali degli alimenti indicati. La risposta deve essere un numero intero, senza spiegazioni o testo aggiuntivo. In caso di dubbio, restituisci 0.
- **Prompt 4**: Leggi l'elenco degli alimenti e restituisci una stima del totale calorico. Non fornire descrizioni, solo un numero. Se non sei sicuro, rispondi con 0.

4.6.4 Integrazione con il LLM

Di seguito è riportata la funzione per contattare un LLM, nel caso specifico dell'applicazione, GPT 3.5 Turbo:

```
func getKcal(of food: String) async throws -> Int {
    let query = ChatQuery(
        messages: [
            .init(role: .system, content: "Rispondi solo con il numero intero totale delle calorie degli alimenti forniti, senza aggiungere mai del testo di nessun tipo. Se non ci sono alimenti validi o sei incerto rispondi con 0")!,
            .init(role: .user, content: food)!
        ],
        model: .gpt3_5Turbo
    )

    let result = try await openAI.chats(query: query)

    guard let choice = result.choices.first,
          let message = choice.message.content?.string else {
        throw CustomError.noResponse
    }

    return Int(message) ?? 0
}
```

Lo scopo di questa funzione è permettere all'utente di ottenere il conteggio totale delle kcal assunte durante la giornata utilizzando il linguaggio naturale.

Per contattare il LLM, è stata utilizzata l'apposita libreria, con licenza MIT, di MacPaw [1].

5 Conclusioni

Sviluppare iWalk è stata un'esperienza che mi ha dato l'opportunità di approfondire diversi campi: dalla programmazione per dispositivi mobile, all'analisi dati, fino al machine learning. È stato interessante seguire tutto il percorso, partendo dalla raccolta dei dati fino ad arrivare a integrare un modello predittivo all'interno dell'applicazione.

Far funzionare l'applicazione con un dataset personale ha reso il tutto più concreto, facendomi scontrare con la realtà dei dati veri, spesso mancanti o poco strutturati. È stato necessario pulirli, riempire i buchi e validarli.

Aggiungere un LLM ha dato una marcia in più all'app, rendendola più interattiva e utile per chi la usa.

Alla fine, questo progetto mi ha permesso di mettere in pratica quello che ho studiato, provare soluzioni vere e riflettere su come l'IA possa davvero aiutare nella vita di tutti i giorni.

Riferimenti bibliografici

[1] MacPaw. *OpenAI Swift Library*. `https://github.com/MacPaw/OpenAI`. 2024.