Università degli Studi di Salerno

Dipartimento di Informatica

Laurea Triennale in Informatica

Corso: Fondamenti di Intelligenza Artificiale

IMPETUS

Injury and Performance Estimation on Time-series User Synthetic Gym Dataset

Autore: Alessandro Ambrosio

Matricola: 0512119148

Anno Accademico: A.A. 2025/2026

GitHub Repository

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni il settore del fitness sta integrando in modo crescente strumenti digitali (app, wearable, servizi online) e approcci data-driven per supportare la personalizzazione dell'allenamento e il monitoraggio dei progressi individuali. In questo contesto, la disponibilità di log di allenamento (sessioni, volumi, intensità, aderenza) abilita la costruzione di pipeline di analisi e modelli di machine learning in grado di trasformare i dati in indicatori utili, sia per la progressione della performance sia per la gestione del rischio legato al carico.

Il presente lavoro si focalizza sul modulo **IMPETUS**, sviluppato come componente complementare al modulo STATUS e orientato a compiti di regressione su variabili continue derivate dalla dinamica temporale dell'allenamento. La motivazione nasce dall'esigenza di unire competenze di dominio (programmazione dell'allenamento: carichi, frequenza, aderenza, progressione) e competenze tecniche (data pipeline, feature engineering, training e valutazione di modelli), con l'obiettivo di produrre un flusso riproducibile e documentato end-to-end.

1.1 Obiettivo del modulo IMPETUS

L'obiettivo del modulo è stimare grandezze sintetiche day-level (o aggregate) che descrivano lo stato dell'utente e la sua evoluzione nel tempo, adottando un'impostazione supervisionata e un protocollo di valutazione coerente con la natura temporale dei dati. In particolare, IMPETUS è articolato in due pipeline:

- **Pipeline A** (*Performance Score Regression*): predizione del punteggio di performance su scala 0–10 a partire da feature Banister-like (fitness/fatigue/TSB), indicatori di carico (es. ACWR), trend e lag temporali.

- **Pipeline B** (*Injury Risk Score Regression*): stima di un injury risk score su scala 0–10 per supportare scenari di monitoraggio e analisi *what-if* legati alla gestione del carico (es. variazioni di ACWR).

1.2 Contributi principali

I contributi del modulo IMPETUS possono essere sintetizzati come segue:

- Progettazione di un workflow sperimentale riproducibile: caricamento dati preprocessati, training, valutazione, selezione del modello e salvataggio degli artefatti per inferenza e demo.
- Analisi quantitativa dell'impatto delle feature temporali (lag) tramite ablation study (WITH lag vs WITHOUT lag e ablation mirate).
- Produzione di output di interpretazione (feature importance e controlli di dominanza delle feature) utili per discutere criticamente i risultati e la robustezza del modello nel dominio sintetico.

1.3 Struttura del documento

Il resto dell'elaborato è organizzato come segue: dopo la sezione di Feature Engineering, la **Metodologia sperimentale** descrive i modelli considerati e il protocollo di validazione; i **Risultati** riportano metriche e confronti tra modelli, ablation e analisi di interpretabilità; la **Discussione** interpreta criticamente i risultati e le implicazioni progettuali; infine, **Limitazioni e sviluppi futuri** esplicitano i vincoli del dominio sintetico e i possibili miglioramenti del sistema.

1.4 Specifiche PEAS e analisi dell'ambiente

1.4.1 Specifiche PEAS

P — Performance measure (misura di prestazione). *Definizione.* È la misura con cui si valuta quanto bene l'agente svolge il compito assegnato. *Nel progetto.* La prestazione del sistema viene misurata tramite metriche di regressione sul test set (ad es. R^2, MAE, RMSE) e controlli di generalizzazione (gap train–test), con l'obiettivo di produrre stime accurate e stabili dei punteggi (Pipeline A e Pipeline B).

E — Environment (ambiente). *Definizione.* È l'insieme degli elementi esterni con cui l'agente interagisce e che determinano il contesto del compito. *Nel progetto.* L'ambiente è costituito dai dati (log e feature day-level) che descrivono l'allenamento nel tempo, inclusi indicatori sintetici (es. fitness/fatigue/TSB, ACWR) e l'evoluzione temporale dello stato dell'utente.

A — Actuators (attuatori). *Definizione.* Sono i canali attraverso cui l'agente produce un'azione/uscita che ha effetto sull'ambiente (anche solo informativo). *Nel progetto.* Gli "attuatori" sono le uscite del sistema: predizione dei punteggi (performance score e injury risk score), eventuali segnali/alert (es. rischio elevato), e output per la demo (grafici, dashboard, spiegazioni sintetiche).

S — Sensors (sensori). *Definizione.* Sono i canali attraverso cui l'agente riceve osservazioni dall'ambiente (input percettivi). *Nel progetto.* I "sensori" corrispondono alle feature disponibili in input: serie temporali e aggregazioni (rolling, trend), lag, indicatori di carico e di stato (ad es. ACWR, fitness, fatigue), oltre a eventuali metadati temporali (data) usati per ordinare correttamente le osservazioni.

1.4.2 Analisi dell'ambiente

Le proprietà dell'ambiente aiutano a capire la complessità del problema e a motivare scelte di modeling e validazione. Nella Tabella 1.1 ogni proprietà è definita in modo intuitivo e viene indicata la scelta più coerente per IMPETUS.

Tabella 1.1: Proprietà dell'ambiente: classificazione per IMPETUS.

Proprietà	**IMPETUS**	**Motivazione nel progetto**
Osservabilità	Parzialmente osservabile	I log/feature sono proxy dello stato reale (es. fisiologia, stress, sonno) e non descrivono tutto ciò che influisce su performance/rischio.
Numero di agenti	Multiagente (cooperativo/misto)	Il sistema "interagisce" con utenti che prendono decisioni (aderenza, intensità, recupero); in genere l'obiettivo è cooperativo (supporto), ma il comportamento dell'utente può non essere allineato (misto).
Determinismo	Stocastico	Il processo reale (e spesso anche la simulazione) include variabilità e rumore: l'evoluzione di performance/rischio non è perfettamente prevedibile.
Episodicità	Sequenziale	Lo stato allenante evolve: carico e recupero oggi influenzano fitness/fatigue e quindi performance/rischio nei giorni successivi.
Staticità	Dinamico	I dati rappresentano una dinamica temporale: anche senza azioni esplicite, lo stato cambia con il passare dei giorni (training/detraining).
Discreto vs continuo	Misto (prevalentemente continuo)	Target e molte feature sono continue (score 0–10, fitness, ACWR), mentre il tempo è spesso discretizzato (giorni/settimane).
Noto vs ignoto	Parzialmente ignoto	Nel mondo reale le dinamiche non sono note e vengono apprese dai dati; nel sintetico sono note al generatore, ma il modello le "vede" solo tramite osservazioni (feature).

Implicazione pratica. Dato che l'ambiente è sequenziale, dinamico e parzialmente osservabile, è essenziale adottare split temporali e controlli anti-leakage nelle fasi di feature engineering e valutazione, per evitare stime eccessivamente ottimistiche delle performance.

Capitolo 2

Scopo, contesto e criteri di validità

2.1 Scopo del report IMPETUS

Questo elaborato documenta esclusivamente il modulo **IMPETUS**, ossia un sistema di regressione supervisionata che stima punteggi continui legati a performance e rischio, a partire da log di allenamento e feature ingegnerizzate. Il report ha come obiettivo (i) descrivere una pipeline end-to-end (preprocessing, feature engineering, training, valutazione, salvataggio artefatti) e (ii) analizzare criticamente le performance ottenute e i segnali che guidano il modello (ablation e feature importance), in un contesto controllato e riproducibile.

Fuori scopo. Sono fuori scopo: (a) la classificazione del livello di esperienza (modulo STATUS), trattata in un report separato; (b) qualunque conclusione clinica o decisione medica; (c) una validazione esterna su dati reali, che viene considerata come sviluppo futuro.

2.2 Dataset sintetico e motivazione d'uso

Il dataset utilizzato è sintetico e deriva da un generatore set-level che simula pianificazione delle sessioni, esecuzione dei set (carico, ripetizioni, RPE), dinamiche temporali (fitness/fatigue) e indicatori aggregati (ad esempio ACWR), oltre a metadati utente. L'uso di dati sintetici consente di controllare le distribuzioni dei fenomeni simulati, tracciare il processo generativo e garantire replicabilità degli esperimenti, riducendo vincoli di privacy e disponibilità di dati reali nelle fasi iniziali di sviluppo. Una analisi piu accurta del dataset e' trattata nel suo capitolo dedicato (3).

2.3 Definizione dei target e dei costrutti (Pipeline A e B)

Il modulo IMPETUS è composto da due task di regressione:

- **Pipeline A (Performance Score Regression)**: stima di un punteggio di performance su scala 0–10, interpretato come proxy sintetico dello stato prestativo/condizione dell'utente in funzione dello storico di carico e recupero.
- **Pipeline B (Injury Risk Score Regression)**: stima di un punteggio di rischio infortunio su scala 0–10, interpretato come proxy sintetico di esposizione a rischio legato principalmente alla gestione del carico (es. spike e dinamiche acute/croniche).

Entrambi i target sono definiti rispetto alle assunzioni del modello generativo e non pretendono di coincidere con misure cliniche o biomeccaniche reali; di conseguenza, i risultati vanno interpretati come capacità del modello di apprendere relazioni nel dominio sintetico.

2.4 Minacce alla validità e mitigazioni (IMPETUS)

Questa sezione esplicita le principali minacce alla validità e le mitigazioni adottate, con lo scopo di chiarire cosa è lecito inferire dai risultati nel dominio sintetico.

2.4.1 Validità di costrutto

Minaccia. Possibile disallineamento tra il concetto informale di "performance" o "rischio" e la loro implementazione operativa nel generatore (il target potrebbe catturare solo alcune componenti del costrutto). **Mitigazione.** Il report definisce esplicitamente i target, discute le assunzioni del generatore e include ablation mirate (es. rimozione di lag del target o di feature dominanti) per verificare se il modello stia imparando segnali fisiologici generali o scorciatoie troppo dirette.

2.4.2 Validità interna (leakage e dipendenze temporali)

Minaccia. Con dati temporali e feature lag/rolling, una validazione non time-aware può introdurre contaminazione (informazione futura nel train rispetto al test) e gonfiare artificialmente le metriche. **Mitigazione.** Gli split train/test sono ordinati cronologicamente

e la ricerca di iperparametri per modelli sensibili (es. Ridge) utilizza cross-validation temporale (`TimeSeriesSplit`), preservando l'ordine nel processo di validazione.

2.4.3 Validità statistica (metriche e reporting)

Minaccia. Metriche aggregate possono nascondere failure mode specifici (ad esempio errori maggiori in particolari range del target o instabilità di metriche percentuali con valori piccoli). **Mitigazione.** Oltre a R^2, MAE e RMSE, viene riportata una versione stabilizzata di MAPE (con denominatore minimo) e sMAPE, e vengono analizzate distribuzioni di residui e scatter Predicted vs Actual.

2.4.4 Validità esterna (generalizzabilità)

Minaccia. Un dataset sintetico può non catturare pienamente la complessità del mondo reale (rumore di misura, eterogeneità inter-soggetto, comportamenti non modellati). **Mitigazione.** Le conclusioni sono limitate al dominio sintetico e l'estensione a dati osservati viene trattata come sviluppo futuro (validazione esterna e ricalibrazione del generatore).

2.5 Riproducibilità e tracciabilità degli esperimenti

Gli esperimenti IMPETUS sono resi riproducibili tramite (i) ordinamento temporale e salvataggio degli split, (ii) seed e parametri fissati ove applicabile, (iii) salvataggio degli artefatti di modeling (modello, lista feature, metriche, iperparametri) in file `.pkl` e `.json`. La presenza di metadati (lista feature per pipeline, configurazioni e opzioni come uso di lag) consente audit e tracciabilità tra generator, feature engineering e risultati di training.

Capitolo 3

Dataset sintetico e generatore

3.1 Panoramica

Il presente progetto utilizza un dataset sintetico set-level generato mediante il supporto di *AI generativa*, progettato per simulare lo storico di allenamento di utenti di palestra. Il processo di generazione è stato strutturato per riprodurre in modo controllato le principali caratteristiche statistiche e dinamiche del dominio, includendo sia comportamento (skip/dropout), sia variabili di esecuzione (carichi, ripetizioni, RPE), sia dinamiche temporali (fatica, detraining, variazioni settimanali). L'obiettivo del dataset è fornire un contesto controllato e riproducibile per l'addestramento e la valutazione del modulo *IMPETUS*, che modella la dinamica temporale del training per supportare decisioni settimanali (es. Push/Maintain/Deload) tramite due pipeline supervisionate su target continui: (A) *Performance Score* (0–100) e (B) *Injury Risk Score* (0–100). Si sottolinea che le osservazioni non derivano da dati reali ma sono state create a fini sperimentali; pertanto, i risultati devono essere interpretati nel contesto di dati sintetici, con le relative limitazioni in termini di generalizzabilità a scenari reali. Le assunzioni e le minacce alla validità collegate all'uso di dati sintetici sono discusse nel Chapter 2.

3.2 Fondamenti scientifici del generatore

Il generatore è progettato per produrre pattern temporali e comportamentali plausibili, ispirandosi a modelli e risultati ricorrenti nella letteratura su carico di allenamento, adattamento e aderenza.

3.2.1 Fitness–Fatigue (Impulse–response)

L'andamento di performance/condizione nel tempo può essere modellato come risposta a impulsi di training con due componenti antagoniste: una componente positiva (fitness) e una negativa (fatigue), entrambe con decadimento esponenziale. [1] Questo razionale è coerente con la famiglia di modelli Banister/Calvert e con l'uso di strutture a due stati (fitness e fatigue) come base concettuale per simulare cicli di overload e recupero. [2], [3] Nel contesto di IMPETUS, tali variabili fungono sia da descrittori day-level (feature) sia da base per la costruzione di target continui legati allo stato di forma.

3.2.2 ACWR e rilevazione di spike

Il rapporto Acute:Chronic Workload Ratio (ACWR) è utilizzato come indicatore sintetico di carico acuto rispetto alla preparazione cronica, e spike (incrementi rapidi) sono spesso associati a incremento di rischio di infortunio in letteratura su team sports. [4] Nel generatore, una soglia operativa di spike pari a `ACWR > 1.61` viene adottata per marcare settimane ad alto carico relativo, in linea con la logica di evitare aumenti rapidi e con discretizzazioni che collocano ~ 1.6 al confine superiore della fascia moderata-alta. [4] In IMPETUS, le informazioni `acwr` e `week_type` sono riutilizzate nella fase di trasformazione temporale (day/week-level) per costruire feature di rischio (es. conteggio spike recenti, esposizione a deload, trend del carico).

3.2.3 Progressive overload e plateau per esperienza

Il principio di progressive overload è implementato imponendo tassi di crescita differenti per livello di esperienza (novizi con crescita più rapida, avanzati con crescita più lenta), simulando sia adattamenti iniziali più veloci sia fenomeni di plateau con l'aumentare dell'anzianità di allenamento. In IMPETUS, tali differenze costituiscono un fattore di realismo del dominio sintetico, ma non devono costituire una scorciatoia per i target continui: la costruzione dei target e la selezione delle feature mirano a evitare dipendenze deterministiche da `experience_label` (cfr. Section 3.10).

3.2.4 RPE e calibrazione (RIR-based)

La Rating of Perceived Exertion (RPE) viene trattata come misura di carico interno, con rumore/errore di stima potenzialmente dipendente dall'esperienza dell'utente. [5] Il generatore si ispira all'approccio RPE basato su Repetitions In Reserve (RIR) per

rendere la variabile più interpretabile e coerente con un contesto di resistance training. [5] Nel contesto IMPETUS, RPE e variabili derivate (medie, varianze, trend su finestre mobili) contribuiscono alla modellazione dello stato di forma e alla quantificazione del carico interno.

3.2.5 Pattern di dropout e aderenza

L'aderenza nei contesti non supervisionati (es. palestra) tende a decrescere nel tempo e la frequenza di allenamento è un forte indicatore di dropout; per questo nel generatore sono presenti eventi `skipped` e profili con rischio diverso. [6] In IMPETUS, segnali di aderenza (es. `workout_status`, `p_skip`) sono particolarmente rilevanti come feature temporali e devono essere trattati con attenzione per evitare leakage tramite aggregazioni che includano informazione futura rispetto all'istante predetto.

3.3 Riproducibilità e versionamento

La generazione è parametrizzata da una configurazione (`CFG`) e da un seed fissato (`CFG.seed`).Per evitare dipendenze da stato globale e garantire replicabilità, il generatore istanzia un RNG moderno e lo propaga alle funzioni di campionamento (pattern *pass-the-RNG*).I metadati di generazione (seed, dimensioni, intervalli temporali, distribuzioni aggregate, changelog) sono salvati in `metadata_v2.json`.

Nota implementativa. Nel codice Python si adotta `np.random.default_rng(seed)` per costruire un oggetto `Generator` e campionare in modo deterministico a seed fissato.

3.4 Tabelle esportate e granularità

Il dataset è organizzato in tabelle CSV correlate; la granularità varia da *utente* a *sessione* a *set* fino a *giorno* (feature derivate). Nel modulo IMPETUS, i dati set/session-level vengono successivamente trasformati in serie temporali day-level e week-level (cfr. Section 3.8).

Tabella 3.1: File esportati e contenuto.

File	Descrizione (chiavi principali)
`users.csv`	1 riga/utente (`user_id`); profilo, label, tratti, finestra temporale
`exercises.csv`	catalogo esercizi (`exercise_id`); split e gruppi muscolari
`workouts.csv`	1 riga/sessione; stato `done/skipped`, probabilità skip, tag split
`workout_plan.csv`	prescrizione per esercizio in sessione; set pianificati, reps range, rest, RIR
`workout_sets.csv`	**primary**; 1 riga/set (`set_id`); carico, reps, RPE, ACWR, `workout_id`
`banister_meta.csv`	parametri Banister per utente (`user_id`): $\tau_F, \tau_D, \beta_F, \beta_D$
`banister_daily.csv`	1 riga/giorno per utente; fitness, fatigue, TSB, performance (0–100)
`metadata_v2.json`	metadati: seed, dimensioni, distribuzioni, changelog

3.5 Configurazione experience-aware

La versione v2.0 introduce una parametrizzazione *experience-aware* (dipendente da `experience_label`) per modellare pattern comportamentali plausibili e differenze sistematiche tra livelli.I parametri principali includono: skip rate di base, calibrazione RPE (bias e varianza), probabilità di settimane spike/deload, durata dello storico, e range di consistency. Nel modulo IMPETUS, tali differenze sono considerate un aspetto del dominio sintetico e non devono determinare i target continui in modo deterministico; per questo la costruzione dei target e il protocollo sperimentale enfatizzano controlli anti-leakage.

Tabella 3.2: Parametri experience-aware principali (v2.0).

Parametro	Beginner	Intermediate	Advanced
Skip p_0 (baseline)	0.13	0.075	0.05
Skip fatigue sensitivity	0.35	0.25	0.15
RPE noise std	1.5	1.0	0.6
RPE bias	+0.5	0.0	-0.2
Spike prob (settimana)	0.20	0.12	0.06
Deload prob (settimana)	0.03	0.05	0.08
Durata storico (giorni)	90–270	240–600	540–1080
Consistency target	0.65–0.80	0.75–0.90	0.85–0.95

3.6 Processo generativo (utente → sessioni → set)

La simulazione è eseguita per utente e scorre le sessioni pianificate tra `start_date` e `end_date`.Ogni sessione può essere completata o saltata (skip), e se completata genera un piano esercizi e una sequenza di set eseguiti.

3.6.1 Pianificazione sessioni

Le date di allenamento sono generate in base a una frequenza settimanale dichiarata (`weekly_freq_declared`) e a un jitter sul giorno della settimana per introdurre variabilità realistica.

3.6.2 Modello di skip (session-level)

La probabilità di skip p_{skip} è definita come una trasformazione logistica della probabilità base per livello p_0 (modulata dalla disciplina utente) e di un termine di fatica saturato:

$$p_{\text{skip}} = \sigma\Big(\text{logit}(p_0) + w_{\text{fat}} \cdot \log(1 + \text{fatigue}) + \epsilon\Big),$$

dove w_{fat} dipende dal livello e ϵ è rumore gaussiano. I campi `p_skip`, `z_skip` e `fatigue_term` sono salvati in `workouts.csv` per consentire audit e analisi.

3.6.3 ACWR settimanale e week type

A inizio di ciascuna settimana utente, viene calcolato un moltiplicatore ACWR che determina il tipo settimana `normal/spike/deload` e influenza l'ampiezza dell'allenamento (es. set eseguiti).Il tipo settimana è salvato a livello set (`week_type`) insieme al valore `acwr`.

3.6.4 Esecuzione set: load, reps, RPE e missingness

Per ciascun esercizio pianificato, il numero di set eseguiti è una funzione dei set pianificati, del profilo utente (`profile`) e del moltiplicatore settimanale.Ogni set produce:

- `load_intended_kg` e `load_done_kg` (con quantizzazione e rumore osservazionale);
- `reps_target` e `reps_done` (con degradazione dovuta alla fatica intra-sessione);
- `rpe_done` (osservato) ottenuto da una componente "true" e da un errore di reporting (bias+varianza) dipendente dal livello.

Per aumentare realismo, è introdotta missingness controllata su `load_done_kg`, `rpe_done` e `feedback`.

3.7 Feature day-level: modello di Banister

A partire dall'impulso giornaliero (`impulse`), viene calcolata una serie temporale day-level tramite un modello impulso-risposta: fitness e fatigue sono ottenute come convoluzione dell'impulso con kernel esponenziali, parametrizzati per utente da τ_F e τ_D.La quantità `TSB` è definita come differenza tra fitness e fatigue, e la variabile `performance` è una trasformazione scalata in [0,100] per interpretabilità. Nel contesto IMPETUS, `fitness`, `fatigue` e `TSB` risultano strutturalmente correlate; la fase di feature engineering e modellazione include quindi controlli e mitigazioni per multicollinearità (cfr. capitolo Feature Engineering di IMPETUS).

3.8 Derivazioni temporali per IMPETUS (day/week-level)

Sebbene il generatore esporti tabelle set/session/day-level, IMPETUS richiede una rappresentazione esplicita della dinamica temporale per utente.A tal fine, i dati vengono trasformati in (i) serie day-level con griglia completa (*inclusi rest days*) e (ii) aggregazioni week-level per supportare decisioni settimanali.

3.8.1 Griglia completa e rest days

Per ciascun utente viene costruita una griglia continua di date nell'intervallo osservato, su cui vengono riportate sia le giornate con training (`workout_status = done`) sia le giornate senza training (rest/skip).Questa scelta evita bias dovuti a serie sparse e permette di calcolare feature su finestre mobili in modo coerente.

3.8.2 Aggregazioni giornaliere e merge

Le tabelle set-level e workout-level vengono aggregate su chiave (`user_id`, `date`) per ottenere statistiche giornaliere (es. volume, intensità, RPE medio, indicatori di spike) e poi merge-ate con `banister_daily.csv` (`fitness`, `fatigue`, `TSB`, `performance`).

3.8.3 Finestre mobili, lag e trend

Sulla serie day-level vengono calcolate rolling windows (7/14/28 giorni) e feature temporali come:

- medie e deviazioni standard su finestre mobili (carico esterno/interno);
- conteggi di eventi recenti (es. giorni di training, settimane spike);
- lag (es. valori a 1/7/14 giorni) e feature di trend (pendenze via regressione lineare locale).

Queste trasformazioni sono implementate rispettando il vincolo causale: ogni feature al tempo t utilizza solo informazione disponibile fino a t (evitando look-ahead).

3.9 Consistency score (post-processing)

La consistency è calcolata come rapporto tra giorni di allenamento completati e giorni totali nella finestra osservata, quindi viene riportata in un intervallo target dipendente dal livello, con lo scopo di evitare distribuzioni degeneri (es. consistency sempre pari a 1.0). In IMPETUS, `consistency_score` può essere utile per audit descrittivo e per interpretare pattern comportamentali; l'uso come feature di training è valutato con cautela per ridurre scorciatoie legate al livello.

3.10 Validazioni e vincoli di qualità

Prima del salvataggio vengono eseguiti controlli automatici per garantire coerenza minima del dataset:

- integrità delle chiavi (`user_id` presente, `workout_id` non nullo nei set);
- vincoli su range (`load_done_kg` in [2.5, 200], `rpe_done` in [1,10], `reps_done` in [1,50]);
- presenza delle 3 classi `experience_label` nel dataset finale;
- stima e verifica della skip rate aggregata per livello.
- `consistency_score` è generata in modo deterministico per livello e viene usata solo per audit descrittivo; l'eventuale inclusione tra le feature di IMPETUS è sottoposta a controlli anti-leakage.

Nota su leakage per IMPETUS. Poiché IMPETUS è un modulo temporale supervisionato, la validazione include controlli aggiuntivi nella fase di costruzione delle serie (day/week-level): (i) assenza di feature che contengano informazione futura rispetto

al target, (ii) separazione tra variabili di dominio (`experience_label`) e target continui, e (iii) audit delle correlazioni tra feature strutturalmente dipendenti (es. `fitness`, `fatigue`, `TSB`).

3.11 Subset stratificato per sviluppo (IMPETUS)

Per velocizzare EDA e iterazioni di feature engineering, viene creato un subset stratificato di utenti bilanciato per classe (es. 170 utenti per livello, 510 totali).Il subset include versioni filtrate delle tabelle principali (suffix `_sampled`) e viene esportato anche in formato ZIP per uso locale.

Capitolo 4

Formulazione del problema

4.1 Contesto e motivazione

Questo progetto nasce dall'interesse per l'integrazione tra programmazione dell'allenamento e strumenti di data science, con l'obiettivo di trasformare log di allenamento in indicatori quantitativi utili per supportare decisioni e personalizzazione. Nel dominio del fitness, il monitoraggio del carico e dello stato dell'atleta può essere formalizzato tramite feature temporali (rolling, trend, lag) e costrutti sintetici (es. fitness/fatigue, ACWR), rendendo naturale l'applicazione di modelli supervisionati per la stima di score continui e interpretabili.

4.2 Problema e obiettivo

Problema. Dato uno storico temporale di allenamento di un utente (sessioni e set eseguiti, volume/intensità, aderenza) e un insieme di feature day-level ingegnerizzate, stimare automaticamente variabili continue che descrivano (i) la condizione prestativa e (ii) l'esposizione a rischio legata a dinamiche di carico.

Obiettivo. Progettare e validare una pipeline end-to-end per due task di regressione:

- **Pipeline A** (*Performance Score Regression*): predire `performance_score_0_10` su scala 0–10 a partire da feature Banister-like, indicatori di carico e feature temporali (trend/rolling/lag).
- **Pipeline B** (*Injury Risk Score Regression*): predire `injury_risk_score_0_10` su scala 0–10 con particolare attenzione alla relazione con feature di carico (es. `acwr`) e alla possibilità di un modello compatto per la demo.

L'obiettivo sperimentale include anche interpretabilità e controllo di scorciatoie: ablation study (WITH vs WITHOUT lag e rimozione di feature dominanti) e analisi di importanza delle feature.

4.3 Assunzioni e vincoli

Si assume che le feature ingegnerizzate catturino in modo sufficiente lo stato allenante e le sue variazioni nel tempo (carico acuto/cronico, trend di fitness, recupero), e che la definizione dei target sia coerente con le assunzioni del generatore sintetico. Il protocollo di valutazione deve rispettare la natura temporale dei dati (split cronologico, nessun uso di informazione futura nella costruzione delle feature) e mantenere un livello di overfitting controllato tramite analisi del gap train–test e ablation mirate. Inoltre, l'output del sistema deve essere riutilizzabile in inferenza e in una demo applicativa, richiedendo salvataggio di artefatti (modello, feature list, metadati e metriche) e coerenza tra training e prediction.

4.4 Criteri di successo

Il modulo IMPETUS è considerato soddisfacente se:

- ottiene performance robuste sul test set temporale (metriche R^2, MAE, RMSE) con gap train–test contenuto;
- mostra coerenza rispetto alle assunzioni di dominio (es. la presenza di segnali fisiologici nelle feature e una relazione interpretabile con indicatori di carico);
- produce risultati interpretabili e auditabili (feature importance, ablation study) che permettono di motivare la selezione del modello e diagnosticare eventuali scorciatoie o dominanze eccessive;
- rilascia artefatti riutilizzabili per inferenza e integrazione in un prototipo dimostrativo.

Capitolo 5

Pre EDA: Target fix

5.1 Pre-EDA: costruzione e *target fix* (IMPETUS)

Questa sezione documenta il notebook `IMPETUS_Target_Fix.ipynb`, utilizzato per costruire e validare tre variabili day-level derivate dal dataset Banister (`banister_daily_sampled.csv`): `acwr`, `performance_score_0_10` e `injury_risk_score`. L'obiettivo è ottenere target/indicatori coerenti, riproducibili e privi di artefatti dovuti a warm-up delle rolling windows o a merge non controllati con le etichette di esperienza.

5.1.1 Input e granularità

Il notebook opera sulla tabella `banister_daily_sampled.csv` (1 riga per utente-giorno), contenente `impulse`, `fitness`, `fatigue`, `TSB` e `performance`. Nel generatore, `performance` è già scalata in $[0, 100]$ per utente (min-max sulla serie fitness-fatigue) e viene mantenuta come variabile informativa separata.

5.1.2 ACWR day-level e gestione warm-up

Si definisce l'Acute:Chronic Workload Ratio (ACWR) su base giornaliera come rapporto tra carico acuto e carico cronico:

$$ACWR_t = \frac{Acute_t}{Chronic_t} \quad (1)$$

dove $Acute_t$ è la media mobile di `impulse` su 7 giorni e $Chronic_t$ è la media mobile su 28 giorni. Per evitare bias nei primi giorni (assenza di storia sufficiente), si introduce

il flag `acwr_valid`, vero solo quando entrambe le rolling windows sono disponibili (min_periods=7 e min_periods=28). I giorni non validi mantengono `acwr` a `NaN`.

5.1.3 Soglie interpretative (Gabbett 2016)

A livello interpretativo (non come etichetta clinica), si adotta la convenzione: *sweet spot* ACWR $\approx$ 0.8–1.3 e *danger zone* ACWR $\geq$ 1.5. Queste soglie sono usate per audit e per definire una funzione graduale di rischio, non per inferire causalità o probabilità reale di infortunio.

5.1.4 Performance score normalizzato (0–10)

Si introduce un indicatore sintetico `performance_score_0_10` (scala 0–10) basato su normalizzazione robusta (percentili 5–95) di `TSB` e `fitness`, combinati con pesi fissi:

$$score = 10 \cdot (0.6 \cdot TSB_{norm} + 0.4 \cdot fitness_{norm}) \quad (2)$$

Questo score è distinto da `performance` (0–100) già presente nel dataset Banister.

5.1.5 Injury risk score (0–10) e definizione piecewise

Si definisce un indicatore continuo `injury_risk_score` su scala 0–10, come mapping euristico piecewise dell'ACWR, con U-shape leggera: (i) penalità bassa per ACWR $<$ 0.8 (underprepared), (ii) fascia 0.8–1.3 (sweet), (iii) fascia 1.3–1.5 (caution), (iv) fascia $\geq$ 1.5 (danger). Per coerenza metodologica, `injury_risk_score` viene calcolato solo quando `acwr_valid` è vero; nei giorni warm-up resta `NaN`.

5.1.6 Merge della label di esperienza e implicazioni

La colonna `experience_label` è un attributo user-level e viene aggiunta tramite merge da `users_sampled.csv`. La distribuzione per *utenti* è bilanciata per costruzione (170 per classe), mentre la distribuzione per *righe-giorno* può risultare sbilanciata perché la durata dello storico (`durationdays`) nel generatore è stratificata per livello di esperienza (storici più lunghi per utenti Advanced).

5.1.7 Validazioni implementate nel notebook

Il notebook include controlli automatici per: (i) percentuale di giorni con `acwr_valid=True`, (ii) range e statistica descrittiva di `acwr` e `injury_risk_score` sui soli giorni validi, (iii) distribuzione per zone ACWR, (iv) controllo anti-leakage per-utente (aggregazione user-level e correlazione tra rischio medio e codifica ordinale dell'esperienza entro soglie conservative). Queste validazioni hanno funzione di audit (qualità del target), non sostituiscono una validazione esterna su dati reali.

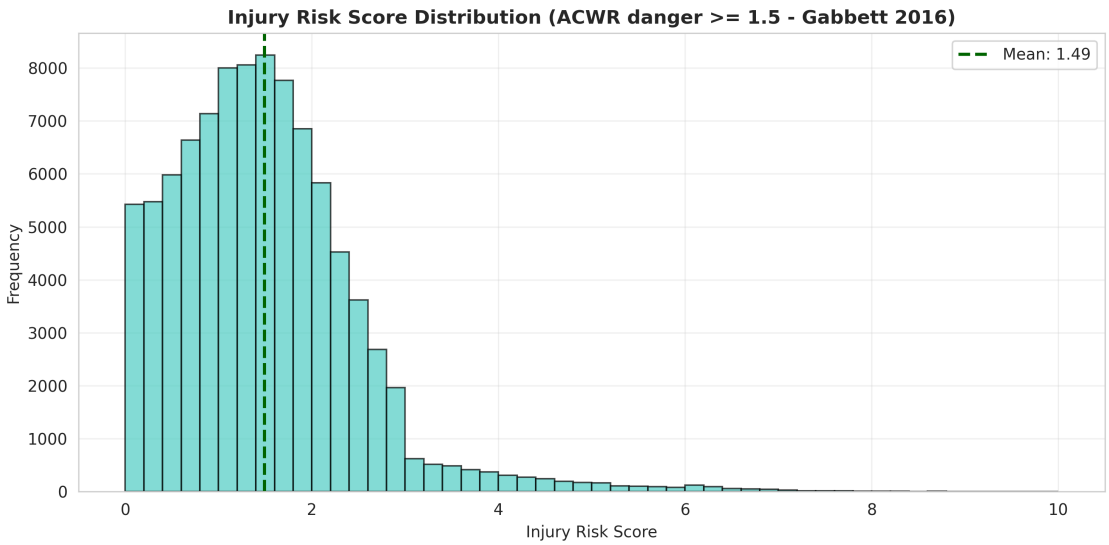

Figura 5.1: Distribuzione di `injury_risk_score` (solo giorni con `acwr_valid=True`).

5.1.8 Output e tracciabilità

L'output del notebook è il file `banister_daily_sampled_fixed.csv`, che conserva le colonne Banister originali e aggiunge: `acwr_valid`, `acwr`, `performance_score_0_10`, `injury_risk_score`, `experience_label`. Le colonne temporanee di calcolo (acute/-chronic load, variabili di debug) vengono rimosse prima del salvataggio.

Capitolo 6

Exploratory Data Analysis (EDA)

6.1 Exploratory Data Analysis (EDA) – IMPETUS

6.1.1 Obiettivi dell'EDA

Questa sezione presenta l'Exploratory Data Analysis (EDA) del dataset day-level `banister_daily_sampled_fixed.csv`. Gli obiettivi sono: (i) profilare dimensioni, periodo temporale e distribuzioni; (ii) validare qualitativamente e quantitativamente i due target continui (Pipeline A e Pipeline B); (iii) individuare pattern temporali e relazioni feature–target utili alla successiva fase di Feature Engineering (FE).

6.1.2 Profilazione del dataset e note di granularità

Il dataset è a granularità *user-day* (una riga per utente e giorno) e contiene feature derivate dal modello Fitness–Fatigue (Banister), tra cui `fitness`, `fatigue`, `TSB` e `performance` (0–100). Nel subset utilizzato per sviluppo, la distribuzione della variabile `experience_label` è bilanciata a livello *utente* (stesso numero di utenti per classe), ma non necessariamente a livello di *righe-giorno*: la durata dello storico `durationdays` è experience-aware, quindi utenti Advanced tendono ad avere più giorni osservati rispetto ai Beginner.

6.1.3 ACWR e gestione warm-up (`acwr_valid`)

L'ACWR è definito come rapporto tra carico acuto e carico cronico, implementati come medie mobili dell'`impulse` su 7 e 28 giorni: `ACWR = acute_load(7d) / chronic_load(28d)`. Poiché la metrica richiede la disponibilità di entrambe le finestre

rolling, nei primi giorni della serie (warm-up) ACWR non è definibile; per evitare bias, si introduce `acwr_valid` e tutte le analisi su ACWR e su risk vengono condotte solo sul subset con `acwr_valid=True`. La distribuzione dell'ACWR e la spike detection (soglia danger) sono visualizzate in Figura 6.1.

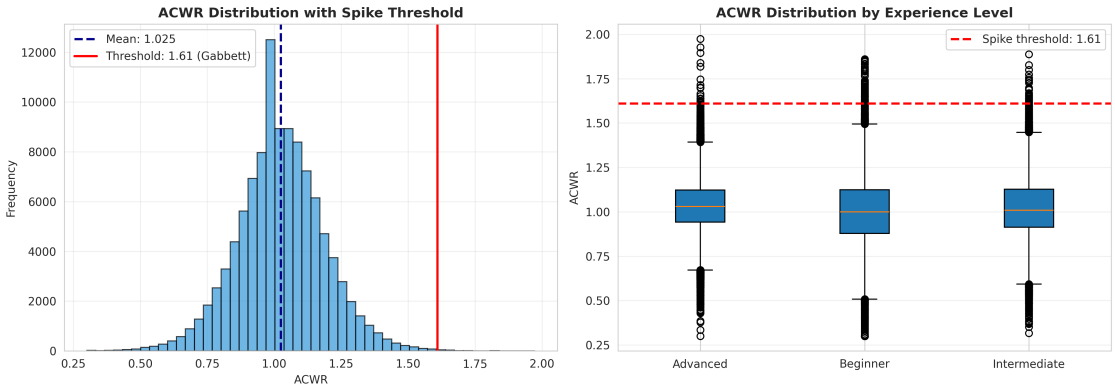

Figura 6.1: Distribuzione di ACWR e zone interpretative (solo giorni con `acwr_valid=True`); confronto per `experience_label`.

6.1.4 TSB (Training Stress Balance) e dinamiche Banister

Il `TSB` è definito come differenza tra `fitness` e `fatigue` (`TSB = fitness - fatigue`) e rappresenta un indicatore sintetico di equilibrio tra adattamento e affaticamento nel modello Fitness–Fatigue. L'EDA verifica la distribuzione del TSB, la quota di giorni con TSB positivo/negativo e differenze per esperienza; tali risultati sono visualizzati in Figura 6.2.

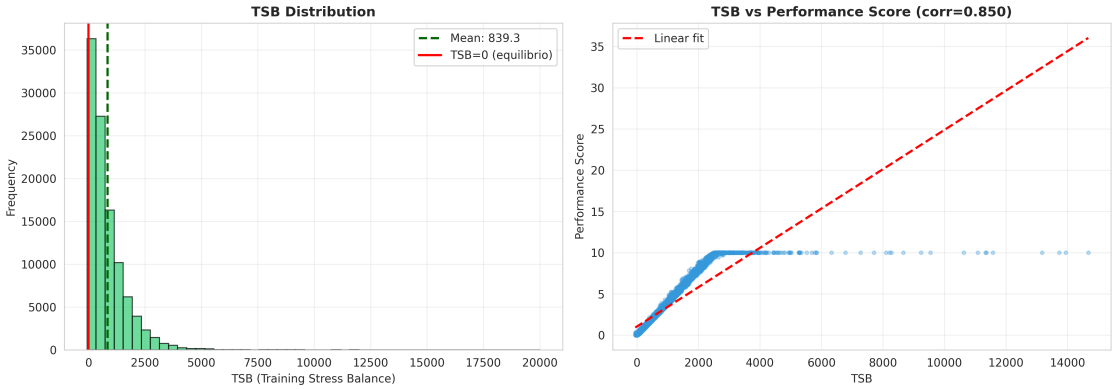

Figura 6.2: Distribuzione di `TSB` e statistiche descrittive per `experience_label`.

6.1.5 Pipeline A: target `performance_score_0_10`

La Pipeline A utilizza `performance_score_0_10` come score sintetico di prontezza/-performance su scala 0–10, costruito come combinazione pesata di `TSB` e `fitness` (normalizzati su quantili 5–95 per robustezza agli outlier). Essendo il target derivato direttamente da variabili Banister, è atteso osservare correlazioni elevate con `TSB` e `fitness`; l'EDA verifica range, distribuzione e comportamento per livello di esperienza. La distribuzione del target e il confronto per esperienza sono riportati in Figura 6.3.

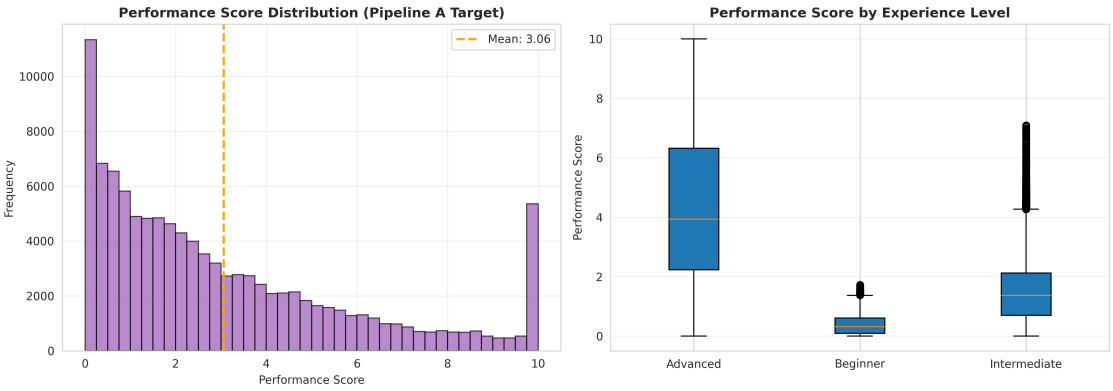

Figura 6.3: Distribuzione di `performance_score_0_10` (target Pipeline A) e confronto per `experience_label`.

6.1.6 Pipeline B: target `injury_risk_score` (solo validi)

La Pipeline B utilizza `injury_risk_score` come score sintetico di rischio (0–10) calcolato esclusivamente sui giorni con `acwr_valid=True`. Il target è definito come funzione continua piecewise dell'ACWR basata su zone operative (under/sweet spot/caution/-danger) e soglia di danger, coerente con la logica di spike detection già utilizzata nel generatore. Di conseguenza, una forte associazione tra `acwr` e `injury_risk_score` è attesa per costruzione; l'EDA verifica distribuzione, range e assenza di dipendenza spuria dall'esperienza mediante aggregazione per utente. La distribuzione del rischio e la sua relazione con l'ACWR sono riportate in Figura 6.4.

6.1.7 Correlazioni feature–target e collinearità attesa

Si calcola una matrice di correlazione tra feature Banister e target (su righe con ACWR valido), utile a supportare la selezione delle feature per ciascuna pipeline. È attesa collinearità elevata tra `fitness`, `fatigue` e `TSB`, poiché derivano dallo stesso processo

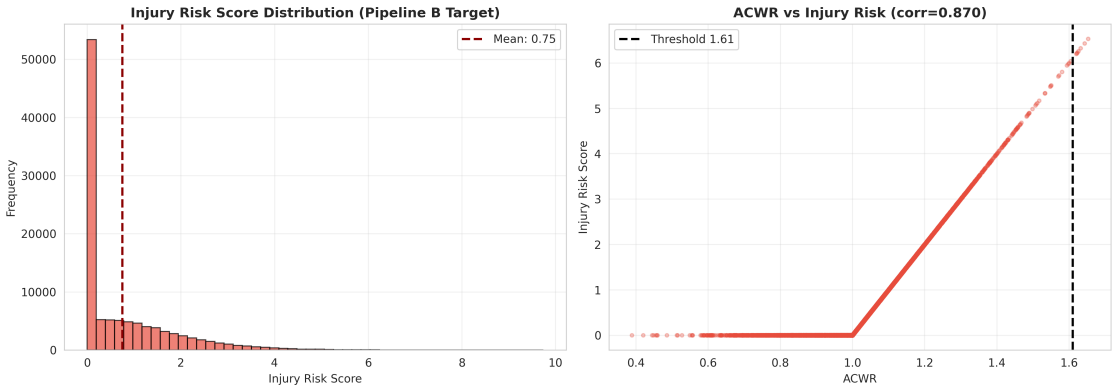

Figura 6.4: Distribuzione di `injury_risk_score` (solo giorni validi) e relazione con `acwr`.

Fitness–Fatigue e includono relazioni deterministiche (ad esempio `TSB = fitness - fatigue`). La matrice completa è riportata in Figura 6.5.

6.1.8 Pattern temporali (time series) su utenti campione

Per ispezionare dinamiche nel tempo e coerenza qualitativa delle serie, vengono tracciati utenti campione (uno per livello di esperienza) con finestre temporali differenti. L'obiettivo è osservare: (i) stabilità e variabilità di `TSB`, (ii) andamento di `performance_score_0_10`, (iii) eventuali giorni in zona danger ACWR e la coerenza con `injury_risk_score` nei soli giorni validi. Le serie temporali sono riportate in Figura 6.6.

6.1.9 Data quality checks e note di interpretazione

Si eseguono controlli finali su: duplicati (`user_id`, `date`), range dei target (0–10) e consistenza del vincolo `TSB = fitness - fatigue`. I missing su `acwr` e `injury_risk_score` sono interpretati come missingness attesa da warm-up (`acwr_valid=False`) e vengono gestiti tramite subset esplicito nelle analisi e nelle validazioni.

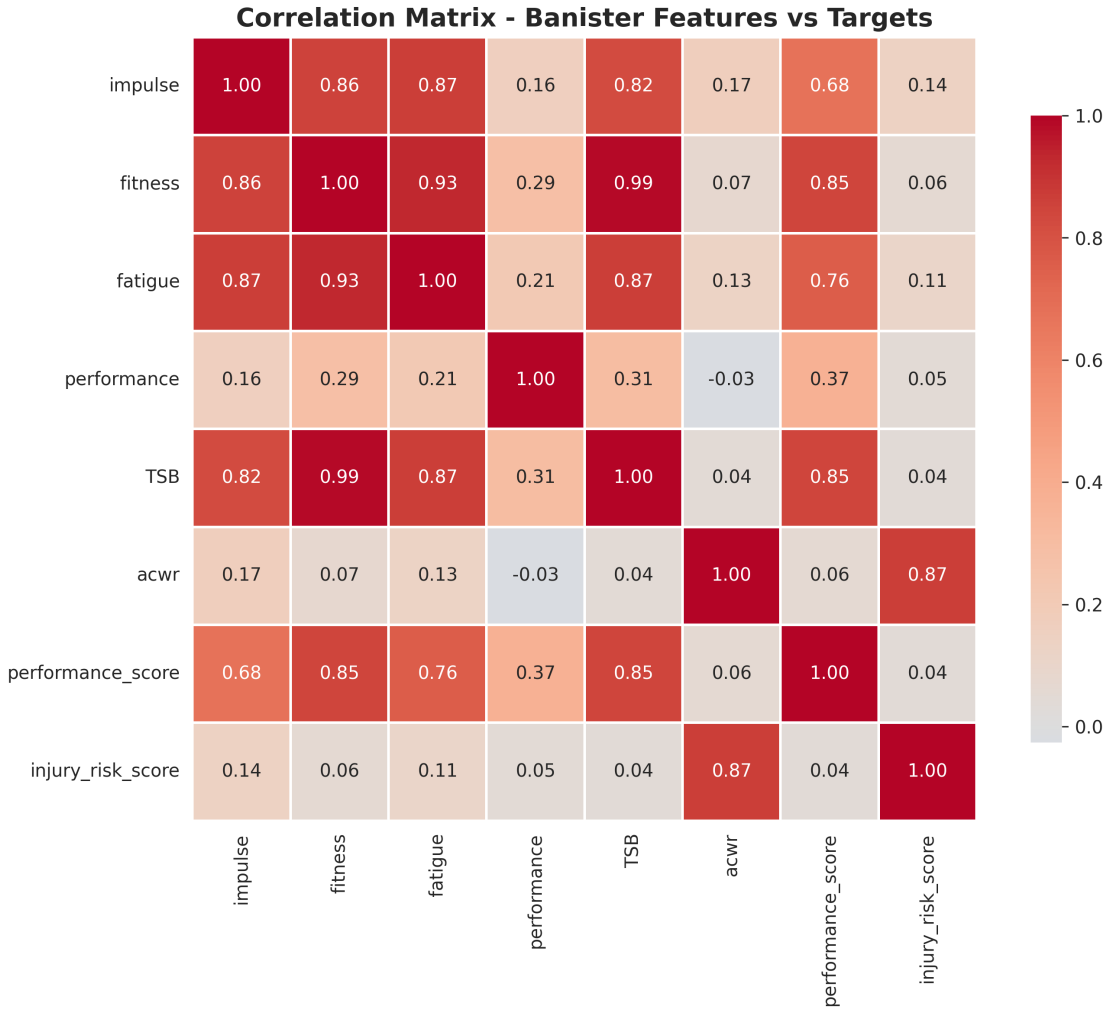

Figura 6.5: Matrice di correlazione tra feature Banister e target su righe con `acwr_valid=True`.

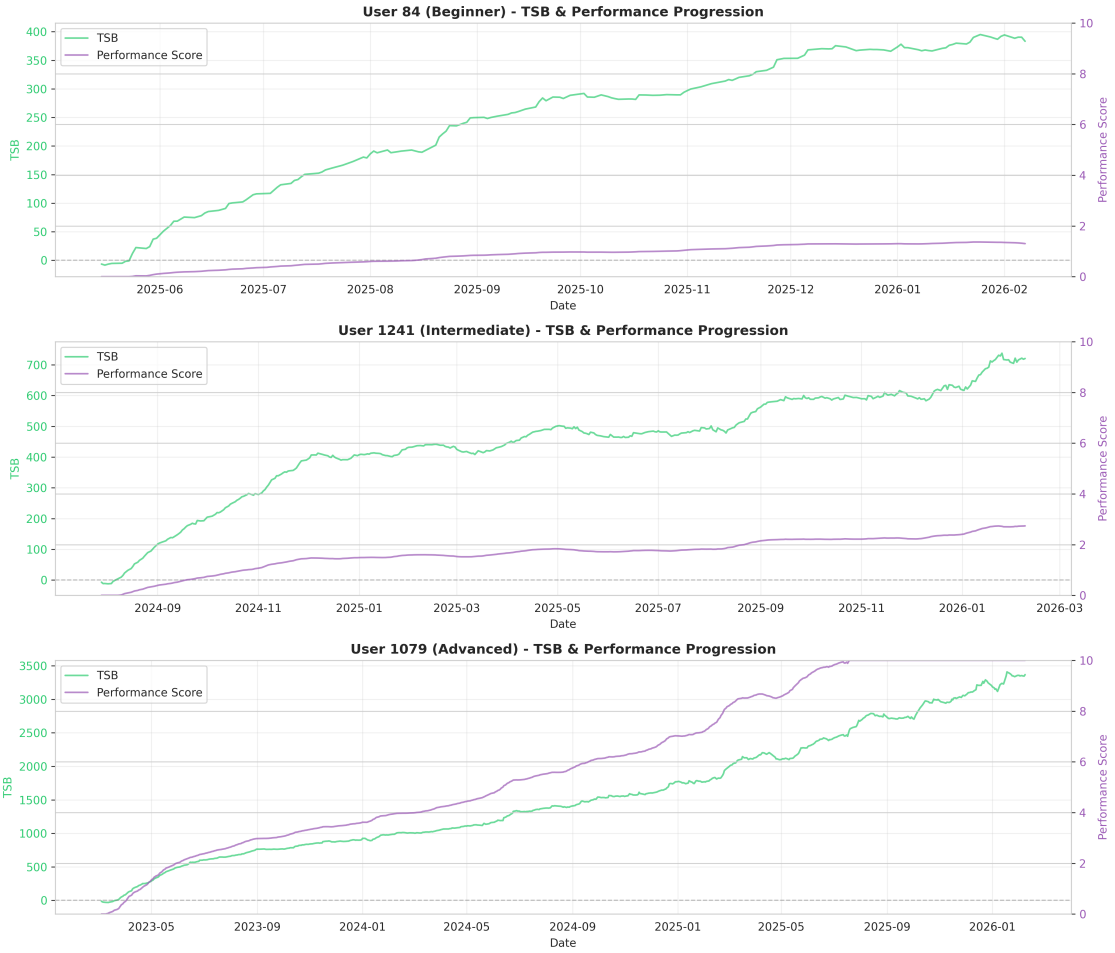

Figura 6.6: Pattern temporali su utenti campione: `TSB`, `acwr` (validi) e target delle due pipeline nel tempo.

6.1.10 Output dell'EDA e implicazioni per la Feature Engineering

Dall'EDA emerge una chiara separazione dei driver delle due pipeline: (i) `TSB`/`fitness` dominano la Pipeline A per costruzione del target; (ii) `acwr` domina la Pipeline B per costruzione del risk score. La FE si focalizzerà su feature temporali (rolling, lag, trend) e su split temporali corretti (no shuffle) per preservare causalità e coerenza in scenari time-series.

Capitolo 7

Feature Engineering

7.1 Feature Engineering (FE) – IMPETUS

7.1.1 Obiettivo e output della pipeline

Questo capitolo descrive la pipeline di Feature Engineering (FE) per il modulo IMPETUS, finalizzata alla costruzione di due task di regressione su scala 0–10: (i) **Pipeline A** per stimare `performance_score_0_10`, e (ii) **Pipeline B** per stimare `injury_risk_score`. La FE opera sul dataset day-level `banister_daily_sampled_fixed.csv`, in cui ogni riga rappresenta un giorno osservato per un determinato utente (`user_id`, `date`). L'output consiste in quattro file separati (train/test per ciascuna pipeline) e negli artefatti di preprocessing (scaler e metadata), in modo da rendere riproducibile la fase di modeling.

7.1.2 Input dataset e granularità (user-day)

Il dataset di input contiene variabili derivate dal modello Fitness–Fatigue (Banister) e indicatori di carico, tra cui `impulse`, `fitness`, `fatigue`, `performance` e `acwr`. Sono presenti inoltre due colonne target (`performance_score_0_10` e `injury_risk_score`) e una colonna categoriale di contesto (`experience_label`). La colonna `acwr_valid` marca i giorni per cui l'ACWR è definibile (warm-up completato), e viene utilizzata per filtrare in modo coerente la Pipeline B.

7.1.3 Feature set di base (Banister + carico)

Come baseline, la pipeline include un set di feature "raw" presenti nel day-level: `impulse`, `tau_F`, `tau_D`, `beta_F`, `beta_D`, `fitness`, `fatigue`, `performance`, `acwr`. Queste variabili

rappresentano: (i) lo stimolo giornaliero (`impulse`), (ii) parametri utente del modello (`tau_*`, `beta_*`), (iii) stati latenti Banister (`fitness`/`fatigue`), e (iv) una misura sintetica di spike/load ratio (`acwr`). Durante la FE è stata rilevata varianza nulla per `beta_F` e `beta_D` nel training set, per cui tali feature vengono rimosse automaticamente prima dello scaling.

7.1.4 Rolling windows (contesto temporale)

Per catturare contesto a breve/medio/lungo termine, vengono costruite rolling windows (medie mobili) su 7, 14 e 28 giorni per un subset di variabili chiave: `fitness`, `impulse`, `acwr` (e inizialmente `TSB`). Le finestre a 7 e 14 giorni utilizzano `min_periods=1` per garantire valori anche all'inizio della storia, mentre la finestra a 28 giorni utilizza `min_periods=7` per ridurre instabilità numerica su finestre molto corte. In aggiunta, viene calcolata la volatilità di ACWR tramite deviazione standard rolling (`acwr_rolling_7d_std` e `acwr_rolling_14d_std`) come proxy di instabilità/variabilità del carico.

7.1.5 Lag features (history e setting autoregressivo)

Vengono calcolate lag features a 7 e 14 giorni per catturare la storia recente delle variabili: `fitness`, `fatigue`, `acwr` (e inizialmente `TSB`). In questa implementazione si adotta esplicitamente un **setting autoregressivo** (opzione `USE_TARGET_LAGS=True`): vengono inclusi anche i lag a 7 e 14 giorni dei target (`performance_score_0_10_lag_7d`, `injury_risk_score_lag_7d`, e analoghi a 14 giorni). Questa scelta è coerente con uno scenario di forecasting in cui lo score storico è disponibile come variabile osservata e può migliorare stabilità predittiva su serie smooth; non viene quindi interpretata come leakage, ma come parte del design del problema.

7.1.6 Trend features (delta, slope, acceleration)

Per catturare progressioni e cambi di regime, vengono costruite feature di trend: (i) delta su 28 giorni per `fitness` e `impulse` (`delta_fitness_28d`, `delta_impulse_28d`); (ii) slope su 14 giorni per `fitness` (`fitness_trend_14d`); (iii) accelerazione ACWR a 7 giorni (`acwr_acceleration_7d`). L'accelerazione di ACWR è definita come differenza rispetto al valore di 7 giorni prima e serve a catturare aumenti rapidi (spike dynamics) oltre al livello assoluto di ACWR. Queste feature sono motivate dall'EDA, che mostra

come la variabilità del carico e la variazione nel tempo possano contribuire alla spiegazione di risk e performance.

7.1.7 Interaction terms (sinergie tra forma e carico)

Sono incluse tre interazioni per catturare sinergie non lineari: `TSB_x_ACWR`, `fitness_x_fatigue` e `impulse_x_acwr`. `TSB_x_ACWR` modella un caso intuitivo: carico relativo elevato (ACWR) combinato con uno stato di forma/stress (TSB/fitness-fatigue) può rappresentare situazioni più critiche di carico. `impulse_x_acwr` combina volume/intensità giornalieri con ratio acute/chronic e in EDA risulta particolarmente associato al risk rispetto a `impulse` da solo.

7.1.8 Gestione multicollinearità (drop di TSB e derivati)

Le feature Banister presentano relazioni strutturali (ad esempio `TSB = fitness - fatigue`), e rolling/lag delle stesse variabili risultano inevitabilmente molto correlati. Per una riduzione "ovvia" e interpretabile, si rimuove `TSB` e tutte le sue derivate (`TSB_rolling_*`, `TSB_lag_*`, `delta_TSB_*`, `TSB_trend_*`), mantenendo però l'interazione `TSB_x_ACWR` in quanto non è una semplice copia lineare di una singola variabile. La validazione tramite matrice di correlazione sulle feature ingegnerizzate conferma la presenza di multicollinearità residua (attesa in contesti time-series), che viene trattata come ridondanza informativa monitorata più che come errore da eliminare completamente.

7.1.9 Pulizia dati: warm-up, NaN e coerenza target

Le trasformazioni rolling/lag introducono valori mancanti nei primi giorni della storia per ciascun utente; tali righe vengono eliminate mediante `dropna` sulle feature selezionate. La pipeline costruisce due dataset puliti separati: (A) richiede target `performance_score_0_10` definito, (B) richiede `acwr_valid=True` e target `injury_risk_score` definito, garantendo coerenza con la definizione di risk basata su ACWR valido. Questo passaggio evita che il modeling debba gestire NaN nei target e rende esplicito il dominio temporale effettivamente utilizzato (post warm-up).

7.1.10 Train/test split temporale (global_date) e shift di popolazione

Lo split train/test è eseguito in modo temporale (80/20) sulla colonna `date`, senza shuffle, per preservare causalità e natura time-series del problema. La data di cut è definita come quantile 0.80 delle date nel dataset pulito, ottenendo una split date pari a 2025-11-19, con train fino a tale data e test successivo. Poiché la durata dello storico è experience-aware (utenti Advanced tendono ad avere più giorni osservati), lo split temporale globale induce un marcato **population shift** nella distribuzione di `experience_label` tra train e test (train dominato da Advanced, test più bilanciato); tale scelta viene mantenuta intenzionalmente e va interpretata come scenario di generalizzazione temporale *con* shift di popolazione.

7.1.11 Scaling e rimozione feature costanti

Per uniformare il preprocessing e facilitare l'uso di modelli sensibili alla scala, viene applicato `StandardScaler` alle sole feature numeriche, con **fit sul train** e transform su train+test. Prima dello scaling, vengono eliminate automaticamente le feature a varianza nulla nel train (es. `beta_F`, `beta_D`), in quanto non aggiungono informazione e possono generare colonne costanti dopo la standardizzazione. I target non vengono scalati poiché sono già normalizzati in [0,10] per costruzione.

7.1.12 Artefatti salvati e riproducibilità

Per garantire tracciabilità, la pipeline salva quattro dataset preprocessati: `impetus_train_A.csv`, `impetus_test_A.csv`, `impetus_train_B.csv`, `impetus_test_B.csv`. Vengono inoltre salvati due scaler separati (`impetus_scaler_A.pkl`, `impetus_scaler_B.pkl`) e un file di metadata (`impetus_feature_metadata.json`) contenente lista feature, strategia di split, feature droppate per varianza nulla e forme finali di train/test per ciascuna pipeline. Questi artefatti permettono di replicare il preprocessing in fase di training e di inference, mantenendo coerenza tra notebook di FE e notebook di modeling.

Dettagli completi (appendice). Per evitare di appesantire la trattazione principale, i dettagli operativi della pipeline di Feature Engineering (file di input/output, artefatti salvati e feature list completa post-riduzione) sono riportati in Appendice .4. In particolare, la Tabella 10 riassume l'I/O della pipeline, mentre la Tabella 11 elenca le 35 feature finali utilizzate nel modeling.

Validazione tramite correlazioni (appendice). Come controllo di coerenza della FE, viene calcolata una matrice di correlazione tra un subset rappresentativo di feature ingegnerizzate e i target delle due pipeline. Nei contesti time-series con rolling e lag, correlazioni elevate tra feature "simili" (es. valore corrente vs media mobile vs lag) sono attese e vengono interpretate come ridondanza informativa, non come errore di preprocessing. Le heatmap complete sono riportate in Appendice .4 (Figure 1 e 2).

Capitolo 8

Metodologia sperimentale

8.1 Metodologia sperimentale

8.1.1 Setup dati e split temporale

I dataset per le Pipeline A (performance score) e B (injury risk score) sono stati generati sinteticamente con 68,694 osservazioni di training (dal 2023-05-06 al 2025-11-19) e 17,131 di test (dal 2025-11-20 al 2026-02-08), rispettando un gap di un solo giorno per preservare la causalità temporale.

Per ciascuna pipeline sono state selezionate 35 feature dai metadati `impetus_feature_metadata.json`, escludendo colonne meta (`user_id`, `date`, `experience_label`) e il target stesso. Le feature includono indicatori Banister (`fitness`, `fatigue`, `TSB`, `ACWR`), trend temporali e lag espliciti (`_lag_7d`, `_lag_14d`).

Lo split è stato ordinato cronologicamente e validato con controllo mensile delle osservazioni:

- Pipeline A: distribuzione experience shift (Advanced: 69.5% $\rightarrow$ 36.4%, Beginner: 3.1% $\rightarrow$ 28.0%).
- Pipeline B: target injury risk con media 1.48 (train) vs 1.54 (test), std 0.95 vs 1.11.

8.1.2 Modelli e validazione

Sono stati testati 5 regressori:

- **Ridge** (L2-regularizzato, CV con `TimeSeriesSplit(n_splits=3)` su $\alpha \in \{0.1, 1.0, 10.0\}$) per robustezza a multicollinearità.
- **Decision Tree** (`max_depth=10`, `min_samples_split=50`) per interpretabilità.

- **Random Forest** (`n_estimators=100`, `max_depth=15`) per ensemble bagging.
- **Gradient Boosting** (`n_estimators=100`, `max_depth=5`) per boosting sequenziale.
- **XGBoost** (opzionale, parametri simili a GB).

Metriche di valutazione:

- R^2, MAE, RMSE (primarie).
- MAPE clipped (denominatore minimo 1.0 per stabilità su scala 0–10).
- sMAPE per robustezza su valori piccoli.

Per preservare l'ordine temporale, Ridge usa `TimeSeriesSplit` mentre gli altri modelli sono valutati su train/test puro (già ordinati cronologicamente).

8.1.3 Ablation study

Tre varianti di feature set:

- **Baseline WITH lag**: tutte 35 feature.
- **NO lag**: rimozione di 10 lag (`_lag_7d`, `_lag_14d`), resto 25 feature.
- **Drop target lag** (Pipeline A): rimozione `performance_score_0_10_lag_7d` (e opzionalmente `_lag_14d`).

Capitolo 9

Risultati

9.1 Risultati sperimentali

9.1.1 Pipeline A: Performance Score Regression

Modelli WITH lag

Tabella 9.1: Risultati modelli WITH lag (Pipeline A, 35 feature).

Modello	R^2 Test	MAE Test	RMSE Test	sMAPE Test (%)	Tempo (s)
Ridge (α=0.1)	0.9994	0.038	0.081	2.8	2.9
Decision Tree	0.9997	0.034	0.054	–	4.7
Random Forest	0.9999	0.022	0.037	–	281.6
Gradient Boosting	**0.9999**	**0.018**	**0.028**	**1.1**	293.3
XGBoost	0.9999	0.019	0.031	–	2.1

Best model: Gradient Boosting, $R^2 = 0.9999$, MAE=0.018 (scala 0–10), gap train-test trascurabile ($\sim 5 \times 10^{-6}$).

Ablation study

Interpretazione:

- `performance_score_0_10_lag_7d` domina (importance 0.983): modello fortemente autoregressivo.
- Rimozione lag peggiora significativamente Ridge (multicollinearità), meno GB.
- Banister features (`fitness`, `fatigue`) sufficienti per $R^2 > 0.99$ senza lag.

Tabella 9.2: Ablation lag e target lag (Pipeline A).

Variante	Modello	n. feat.	R^2 Test	MAE Test	sMAPE (%)
WITH lag (35)	Ridge	35	0.9994	0.038	2.8
	GB	35	**0.9996**	0.051	4.2
Drop perf lag 7d (34)	Ridge	34	0.9947	0.094	4.4
	GB	34	0.9991	0.073	5.8
Drop perf lag 7d+14d (33)	Ridge	33	0.7336	0.518	24.0
	GB	33	0.9987	0.087	6.4

Feature importance

Il Gradient Boosting conferma dominanza di `performance_score_0_10_lag_7d` (98.3%), seguita da `fitness` (1.1%) e rolling fitness (0.5%) (Fig. 9.1).

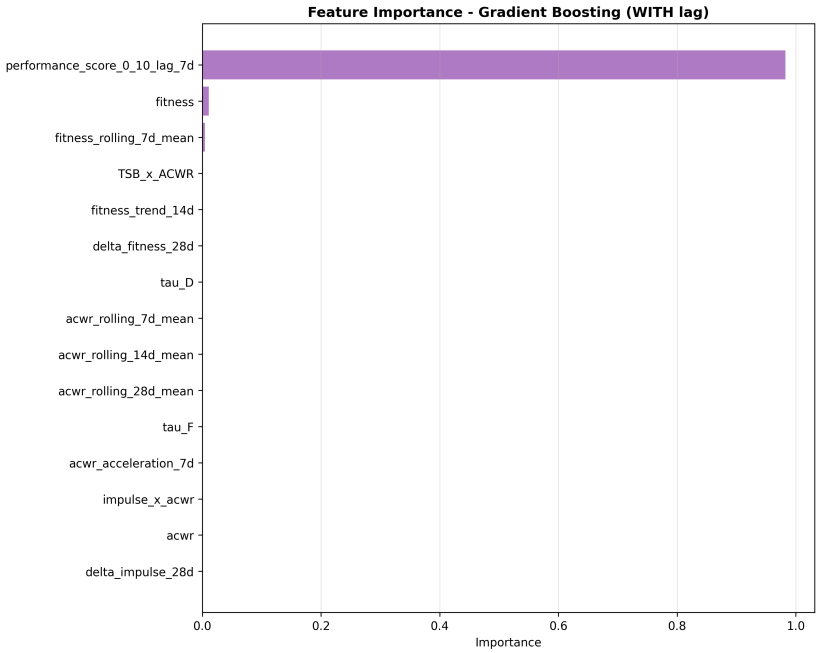

Figura 9.1: Feature importance Gradient Boosting (Pipeline A, WITH lag).

9.1.2 Pipeline B: Injury Risk Score Regression

Modelli WITH lag

Gradient Boosting dominante: $R^2 = 0.999984$, MAE=0.003 (scala 0–10), gap $\sim 5 \times 10^{-6}$.

ACWR dominance

Tabella 9.3: ACWR-only vs full model (Pipeline B).

Modello	R^2 Test	MAE Test	RMSE Test	Features	Tempo (s)
Ridge ACWR-only	0.7790	0.227	0.522	1	<1
DT ACWR-only	0.9996	0.005	0.022	1	5
GB ACWR-only	0.99997	0.004	0.006	1	10
GB full	**0.99998**	**0.003**	**0.004**	35	293

Conclusione: Injury risk è funzione quasi deterministica di ACWR (non lineare). Permutation importance conferma: ACWR drop $R^2 \sim 1.99$, altre feature trascurabili.

9.1.3 Modelli deployati

- **Pipeline A**: `impetus_model_A_with_lag.pkl` (GB, 35 feat.).
- **Pipeline B**: `impetus_model_B_full.pkl` (GB, 35 feat.) + `impetus_model_B_compact_acwr_dt.pkl` (DT, 1 feat. per demo).

Capitolo 10

Discussione

10.1 Considerazioni generali sull'impostazione sperimentale

L'intero workflow è stato impostato con split temporale (train → test in avanti nel tempo), coerente con l'uso di feature laggate e con la natura sequenziale dei segnali di carico/fitness, riducendo il rischio di valutazioni irrealistiche dovute a training su dati futuri e test su dati passati. In particolare, la scelta di una cross-validation `TimeSeriesSplit` per l'ottimizzazione di iperparametri (Ridge) è motivata dal fatto che le tecniche di CV standard non sono appropriate per serie temporali, perché possono contaminare temporalmente i fold.

10.2 Pipeline A (Performance Score): interpretazione dei risultati

La Pipeline A mostra prestazioni elevate con modelli diversi (Ridge, tree-based ed ensemble), suggerendo che la relazione tra feature Banister/trend/rolling e `performance_score_0_10` è fortemente informativa nel dataset sintetico. L'analisi di feature importance evidenzia una dominanza netta della variabile `performance_score_0_10_lag_7d`, e l'ablation mirata conferma che la rimozione di questa feature peggiora sensibilmente le metriche, soprattutto per modelli lineari. Questa evidenza è compatibile con un comportamento *autoregressivo*: il valore passato del target contiene un'informazione diretta sul valore futuro, e il modello sfrutta tale persistenza per ridurre l'errore. La componente fisiologica (fitness/fatigue/ACWR e derivate) rimane comunque rilevante,

poiché nella variante *NO lag* i modelli non lineari mantengono performance molto alte, indicando che lo stato allenante aggregato è sufficiente a spiegare gran parte della variabilità anche senza accesso esplicito alla storia recente del target.

10.3 Pipeline B (Injury Risk): dominanza di ACWR e implicazioni

Nella Pipeline B, i sanity check mostrano che un modello non lineare ACWR-only (Decision Tree / Gradient Boosting) raggiunge prestazioni quasi sovrapponibili al modello full, mentre Ridge ACWR-only resta significativamente più basso, indicando una dipendenza non lineare del rischio rispetto ad ACWR. La permutation importance è coerente con questa lettura: permutare una feature e misurare il calo di score quantifica quanto il modello dipenda realmente da quella variabile, e una dominanza marcata di `acwr` implica che la predizione sia guidata quasi interamente da essa. In termini progettuali, questo rende la Pipeline B particolarmente adatta a una demo *what-if* rapida (variazione di ACWR $\rightarrow$ variazione del rischio), perché il comportamento del modello è stabile e spiegabile con una singola variabile dominante.

10.4 Nota critica su performance molto alte

Prestazioni estremamente elevate (R^2 prossimi a 1 con gap train-test minimo) sono possibili su dataset sintetici quando il target è generato da regole deterministiche o quasi deterministiche, oppure quando alcune feature contengono informazione direttamente/indirettamente legata al target (target leakage). In generale, la data leakage può gonfiare silenziosamente le metriche e produrre una stima troppo ottimistica della generalizzazione, soprattutto in contesti temporali e con feature engineering ricche (rolling/lag). Per questo motivo, i risultati vanno interpretati come validazione della coerenza del generatore e della pipeline di feature engineering, oltre che come performance predittiva sullo split sintetico.

Capitolo 11

Limitazioni e sviluppi futuri

11.1 Limitazioni

- **Natura sintetica dei dati**: la generazione sintetica può non riprodurre completamente complessità, rumore reale, outlier e dipendenze latenti presenti in dati di palestra reali, limitando la trasferibilità dei risultati.
- **Possibile amplificazione di bias**: i processi sintetici possono ereditare o amplificare bias del modello generativo o delle assunzioni di simulazione, con distribuzioni che possono divergere dal mondo reale.
- **Dominanza di feature target-correlate**: in Pipeline A la presenza di `performance_score_0_10_lag_7d` rende il task in parte autoregressivo; in Pipeline B la dominanza di `acwr` rende il rischio quasi deterministico, riducendo l'utilità di modelli complessi rispetto a surrogate più semplici.
- **Rischio di leakage temporale nel feature engineering**: con rolling e lag è essenziale garantire che ogni feature al tempo t dipenda solo da informazioni disponibili fino a t e che la validazione rispetti l'ordine temporale.

11.2 Sviluppi futuri

- **Validazione su dati reali**: integrare un dataset reale (anche piccolo) per verificare stabilità delle relazioni apprese e ricalibrare il generatore sintetico per ridurre mismatch di distribuzione.
- **Raffinamento del generatore del rischio**: introdurre fattori latenti realistici (es. qualità del sonno, stress, storico infortuni) e rumore eteroschedastico per ridurre la determinismo ACWR-only e rendere il task di Injury Risk più informativo.

- **Valutazione causale e robustezza**: inserire test di robustezza (ablation sistematiche, permutation importance) come step standard, poiché quantificano dipendenze reali del modello dalle feature e aiutano a diagnosticare dominanze eccessive.
- **Pipelines end-to-end e prevenzione leakage**: migrare gradualmente a pipeline complete (feature engineering $\rightarrow$ modello) per minimizzare contaminazioni accidentali tra train e test nelle fasi di trasformazione.

Bibliografia

[1] F. Imbach, N. Sutton-Charani, J. Montmain, R. Candau e S. Perrey, «The Use of Fitness-Fatigue Models for Sport Performance Modelling: Conceptual Issues and Contributions from Machine-Learning,» *Sports Medicine - Open*, vol. 8, n. 29, 2022. DOI: 10.1186/s40798-022-00426-x indirizzo: https://pmc.ncbi.nlm.nih.gov/articles/PMC8894528/

[2] T. W. Calvert, E. W. Banister, M. V. Savage e T. Bach, «A Systems Model of the Effects of Training on Physical Performance,» *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 94–102, 1976. DOI: 10.1109/TSMC.1976.5409179 indirizzo: https://pmc.ncbi.nlm.nih.gov/articles/PMC8894528/

[3] E. Banister, T. Calvert, M. Savage e T. Bach, «A Systems Model of Training for Athletic Performance,» *Australian Journal of Sports Medicine*, vol. 7, n. 3, pp. 57–61, 1985. indirizzo: https://pmc.ncbi.nlm.nih.gov/articles/PMC8894528/

[4] L. Bowen, A. S. Gross, M. Gimpel, S. Bruce-Low e F.-X. Li, «Spikes in acute:chronic workload ratio (ACWR) associated with a 5–7 times greater injury rate in English Premier League football players: a comprehensive 3-year study,» *British Journal of Sports Medicine*, vol. 54, n. 12, pp. 731–738, 2019. DOI: 10.1136/bjsports-2018-099422 indirizzo: https://pmc.ncbi.nlm.nih.gov/articles/PMC7285788/

[5] E. R. Helms, J. Cronin, A. Storey e M. C. Zourdos, *Application of the Repetitions in Reserve-Based Rating of Perceived Exertion Scale for Resistance Training*, PMCID: PMC4961270, 2016. indirizzo: https://pmc.ncbi.nlm.nih.gov/articles/PMC4961270/

[6] «Dropout predictors at gyms: a retrospective study,» *Revista Brasileira de Ciência e Movimento (SciELO)*, 2021. DOI: 10.1590/rbce.43.e014220 indirizzo: https://www.scielo.br/j/rbce/a/WtzM3gBFRkcrqY7xKZsVNnm/?lang=en

Appendice A

Dettagli implementativi

.1 Output del generatore

Tabella 1: File generati (7 CSV + 1 JSON)

File	**Formato**	**Granularità / chiave**	**Descrizione (breve)**
users.csv	CSV	1 riga = 1 utente; PK: `user_id`	Anagrafica/attributi utente e label/target (se presenti).
exercises.csv	CSV	1 riga = 1 esercizio; PK: `exercise_id`	Catalogo esercizi (nomi, gruppi muscolari, ecc.).
workouts.csv	CSV	1 riga = 1 workout; PK: `workout_id`, FK: `user_id`	Sessioni eseguite (data, durata, volume aggregato, ecc.).
workout_sets.csv	CSV	1 riga = 1 set; PK: `set_id`, FK: `workout_id`, `exercise_id`	Dettaglio set/reps/load/RPE per ogni workout.
workout_plan.csv	CSV	1 riga = 1 elemento di piano; (chiave da specificare)	Pianificazione allenamenti (planned vs done; utile per `skip_rate`).
banister_daily.csv	CSV	1 riga = 1 giorno-utente; PK: (`user_id`, `date`)	Serie giornaliera per Banister (carico/impulso, fitness/fatigue, ecc.).

File	Formato	Granularità / chiave	Descrizione (breve)
banister_meta.csv	CSV	1 riga = 1 utente; PK: `user_id`	Parametri/meta del modello Banister (costanti, inizializzazioni, ecc.).
metadata_v2.json	JSON	1 documento; campo `version`	Metadati di generazione (seed, range date, conteggi, changelog).

.2 Dizionario delle feature

Questa appendice riporta un data dictionary (codebook) delle variabili presenti nei file generati dal generatore sintetico, includendo definizione, unità, range atteso e note di calcolo.

.2.1 `users.csv`

Tabella 2: Dizionario feature – `users.csv`

Feature	Definizione / calcolo	Unità	Range atteso	Tipo
user_id	Identificativo univoco utente (PK), assegnato come $i + 1$.	id	≥ 1	int
start_date	Data inizio finestra utente, calcolata come `today - duration_days`.	date	ISO (YYYY-MM-DD)	string/date
end_date	Data fine finestra utente, fissata a `today`.	date	ISO (YYYY-MM-DD)	string/date
duration_days	Durata finestra (giorni), stratificata per livello: Beginner 90–270, Intermediate 240–600, Advanced 540–1080.	days	90–1080	int
weekly_freq_declared	Frequenza settimanale dichiarata (clamp e cast), con vincoli 1–6.	sess./week	1–6	int

Feature	Definizione / calcolo	Unità	Range atteso	Tipo
split_type	Tipo split: `PPL` (p=0.7) o `FullBody` (p=0.3).	cat	{PPL, FullBody}	string
profile	Profilo comportamentale: `balanced`, `high_volume`, `high_intensity`, `inconsistent`.	cat	4 categorie	string
experience_label	Etichetta esperienza (ground-truth generatore): Beginner/Intermediate/Advanced.	cat	3 classi	string
experience_latent	Variabile latente continua (campionata uniforme).	score	0–1	float
alpha_adapt	Tasso latente di adattamento.	score	0.03–0.12	float
k_detraining	Tasso latente di detraining.	score	0.005–0.02	float
obs_noise	Rumore osservazionale latente.	score	0.5–1.5	float
resilience	Resilienza latente.	score	0.5–1.2	float
fatigue_sens	Sensibilità alla fatica latente.	score	0.6–1.4	float
rpe_report_bias	Bias individuale di reporting RPE (aggiunto all'RPE osservato).	score	-0.3–0.3	float
discipline	Disciplina (da profilo + rumore, poi clip).	score	0.1–1.0	float
motivation	Motivazione (normale, poi clip).	score	0.3–1.0	float
consistency_score	Aderenza globale: #giorni done/#giorni totali, poi clip nel range target per livello (Beginner 0.65–0.80, Intermediate 0.75–0.90, Advanced 0.85–0.95).	rate	0.65–0.95	float

.2.2 `exercises.csv`

Tabella 3: Dizionario feature – `exercises.csv`

Feature	Definizione	Range atteso	Tipo
exercise_id	Identificativo esercizio (PK) nel catalogo.	≥ 1	int
exercise_name	Nome esercizio (stringa).	–	string

Feature	Definizione	Range atteso	Tipo
target_muscle_group	Gruppo muscolare target (categoria).	categorie finite	string
split_cat	Categoria split: Push/Pull/Legs/FullBody.	categorie finite	string

.2.3 `workouts.csv`

Tabella 4: Dizionario feature – `workouts.csv`

Feature	Definizione / calcolo	Unità	Range atteso	Tipo
workout_id	Identificativo globale workout (PK), assegnato come indice ordinato per (`user_id`, `date`).	id	≥ 1	int
user_id	Identificativo utente (FK $\rightarrow$ `users.user_id`).	id	≥ 1	int
date	Data della sessione pianificata.	date	ISO (YYYY-MM-DD)	string/date
week_index_user	Indice settimana nella carriera utente: $\lfloor (d - start)/7 \rfloor + 1$.	week	≥ 1	int
session_tag	Tag sessione in base allo split: Push/Pull/Legs o FullBody.	cat	set finito	string
workout_status	Stato: `done` oppure `skipped`.	cat	{done, skipped}	string
z_skip	Logit score del modello skip: $z = \text{logit}(p0) + w_f \cdot \log(1 + \text{fatigue}) + \epsilon$.	score	real	float
p_skip	Probabilità di skip: $\sigma(z)$ (sigmoid).	prob	0–1	float
fatigue_term	Termine fatica usato nello skip: $\log(1 + \text{fatigue})$ con cap.	score	≥ 0	float
experience_label	Etichetta esperienza associata all'utente (ridondante rispetto a `users`).	cat	3 classi	string

.2.4 `workout_plan.csv`

Tabella 5: Dizionario feature – `workout_plan.csv`

Feature	Definizione / calcolo	Unità	Range atteso	Tipo
workout_id	Identificativo workout (FK → `workouts.workout_id`), propagato tramite mapping (`user_id`, `date`).	id	≥ 1	int
user_id	Identificativo utente.	id	≥ 1	int
date	Data sessione.	date	ISO (YYYY-MM-DD)	string/date
session_tag	Tag sessione (Push/Pull/Legs/Full-Body).	cat	set finito	string
exercise_id	Identificativo esercizio (FK → `exercises.exercise_id`).	id	≥ 1	int
sets_planned	Set pianificati per esercizio, dipendono da experience (Beg 2–4, Int 3–5, Adv 4–6) e vengono ridotti in caso di infortunio (moltiplicatore 0.6, min 1).	set	1–6	int
reps_min	Min reps pianificate (fisso).	reps	6	int
reps_max	Max reps pianificate (fisso).	reps	12	int
rest_planned_sec	Recupero pianificato tra set.	sec	$\{90,120,150,180\}$	int
rir_target	Target RIR per livello: Beg 2–4, Int 1–3, Adv 0–2.	reps	0–4	int

.2.5 `workout_sets.csv`

Tabella 6: Dizionario feature – `workout_sets.csv` (tabella primaria a livello set)

Feature	**Definizione / calcolo**	**Unità**	**Range atteso**	**Tipo**
set_id	Identificativo set (PK) stringa: `{user}S{counter}` (es. 0001S0000001).	id	unico	string
workout_id	Identificativo workout (FK), propagato da (`user_id`, `date`).	id	≥ 1	int
user_id	Identificativo utente.	id	≥ 1	int
date	Data sessione.	date	ISO (YYYY-MM-DD)	string/date
week_index_user	Indice settimana nella carriera utente.	week	≥ 1	int
week_type	Tipo settimana (ACWR): `normal`, `spike`, `deload`.	cat	3 categorie	string
acwr	Valore ACWR settimanale usato come moltiplicatore (normal circa 0.90–1.25; spike 1.35–1.85; deload 0.65–0.85).	ratio	≥ 0	float
session_tag	Tag sessione (Push/Pull/Legs/FullBody).	cat	set finito	string
exercise_id	Esercizio eseguito (FK).	id	≥ 1	int
set_index	Indice del set all'interno dell'esercizio nella sessione (1..sets_done).	idx	1–10	int
reps_target	Reps target campionate tra `reps_min` e `reps_max`.	reps	6–12	int
reps_done	Reps eseguite: dipendono da target e fatica, clampate (validazione: 1–50).	reps	1–50	int
load_intended_kg	Carico "inteso": q_load(intensity $\cdot$ c_max) con step 0.25 kg.	kg	≥ 0	float

Feature	Definizione / calcolo	Unità	Range atteso	Tipo
load_done_kg	Carico eseguito: combinazione capacity-based e daily-variation, quantizzato (step 0.25 kg); missing con probabilità `p_missing_load`; clamp post-process in [2.5, 200.0].	kg	2.5–200.0 (se non missing)	float/NaN
rpe_done	RPE osservato: mix di intensità, motivazione, fatica; poi noise+bias experience-aware; quantizzato step 0.5 e clamp 1–10; missing con `p_missing_rpe`.	1–10	1–10 (se non missing)	float/NaN
rest_planned_sec	Recupero pianificato (replicato dal piano).	sec	{90,120,150,180}	int
rir_target	RIR target (replicato dal piano).	reps	0–4	int
feedback	Feedback testuale opzionale (5%), missing con `p_missing_feedback`.	text	–	string/None

.2.6 `banister_meta.csv`

Tabella 7: Dizionario feature – `banister_meta.csv`

Feature	Definizione / calcolo	Unità	Range atteso	Tipo
user_id	Identificativo utente (PK e FK).	id	≥ 1	int
tau_F	Costante di tempo Fitness per utente (campionata da normale e troncata a ≥ 7).	days	≥ 7	float
tau_D	Costante di tempo Fatigue per utente (campionata e troncata a ≥ 2).	days	≥ 2	float
beta_F	Guadagno Fitness globale (config).	coeff	> 0	float
beta_D	Guadagno Fatigue globale (config).	coeff	> 0	float

.2.7 `banister_daily.csv`

Tabella 8: Dizionario feature – `banister_daily.csv`

Feature	**Definizione / calcolo**	**Unità**	**Range atteso**	**Tipo**
user_id	Identificativo utente.	id	≥ 1	int
date	Data (giornaliera).	date	datetime	datetime
impulse	Impulso giornaliero: somma sui set del giorno di `load_done_kg` · `reps_done` · `rpe_done`/10 (con contributo 0 se load/RPE missing).	a.u.	≥ 0	float
tau_F	Parametro Banister (replicato da meta per merge).	days	≥ 7	float
tau_D	Parametro Banister (replicato da meta per merge).	days	≥ 2	float
beta_F	Parametro Banister (replicato).	coeff	> 0	float
beta_D	Parametro Banister (replicato).	coeff	> 0	float
fitness	Serie Fitness: convoluzione di `impulse` con pesi esponenziali $\exp(-i/\tau_F)$, scalata per β_F.	a.u.	real	float
fatigue	Serie Fatigue: convoluzione di `impulse` con $\exp(-i/\tau_D)$, scalata per β_D.	a.u.	real	float
TSB	Training Stress Balance: `fitness - fatigue`.	a.u.	real	float
performance	Performance scalata in [0,100] per utente: min-max scaling della serie `fitness - fatigue`.	score	0–100	float

.3 Specifica del file `metadata_v2.json`

Tabella 9: Campi del file `metadata_v2.json`

Campo	**Tipo**	**Significato**
version	string	Versione del generatore/dataset (es. “2.0”).

Campo	Tipo	Significato
date_generated	string (datetime)	Timestamp di generazione del dataset.
seed	integer	Seed random per riproducibilità.
n_users	integer	Numero utenti generati.
n_workouts	integer	Numero workouts totali nel dataset.
n_sets	integer	Numero set totali (righe attese in `workout_sets.csv`).
experience_distribution	object	Conteggio utenti per livello (Beginner/Intermediate/Advanced).
skip_rates	object	Skip rate medio per livello.
date_range.min/max	string (datetime)	Estremi temporali del dataset.
changelog	array[string]	Elenco modifiche e razionali (con riferimenti bibliografici nel testo).

.4 IMPETUS: dettagli Feature Engineering

Tabella 10: I/O della pipeline di Feature Engineering per IMPETUS (Pipeline A e B).

Categoria	Tipo	Path	Note
Input	CSV	`data/synth_set_level_v2/banister_daily_sampled_fixed.csv`	Dataset day-level user-day (510 utenti).
Output (Pipeline A)	CSV	`data/synth_set_level_v2/impetus_train_A.csv`	Train set (split temporale global_date 80/20), target: `performance_score_0_10`.
Output (Pipeline A)	CSV	`data/synth_set_level_v2/impetus_test_A.csv`	Test set (periodo futuro), stesso preprocessing del train.
Output (Pipeline B)	CSV	`data/synth_set_level_v2/impetus_train_B.csv`	Train set, filtrato su `acwr_valid=True`, target: `injury_risk_score`.
Output (Pipeline B)	CSV	`data/synth_set_level_v2/impetus_test_B.csv`	Test set, filtrato su `acwr_valid=True`.
Artefatto	PKL	`models/impetus_scaler_A.pkl`	`StandardScaler` fit su train A, applicato a train+test A.
Artefatto	PKL	`models/impetus_scaler_B.pkl`	`StandardScaler` fit su train B, applicato a train+test B.
Artefatto	JSON	`models/impetus_feature_metadata.json`	Feature list, split strategy, colonne droppate, shape finali.

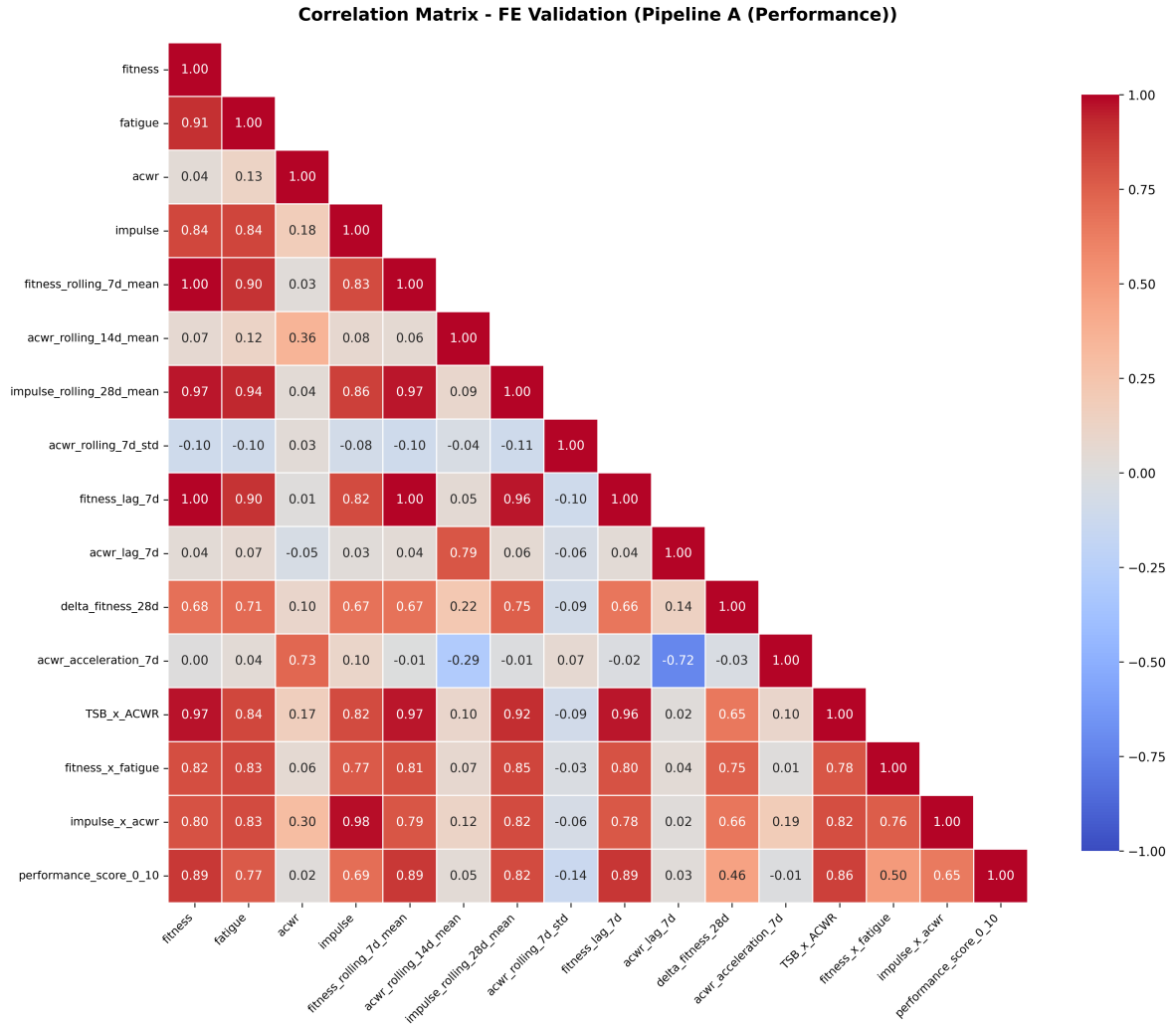

Figura 1: Heatmap correlazioni (Pipeline A): audit FE su subset di feature e target `performance_score_0_10`.

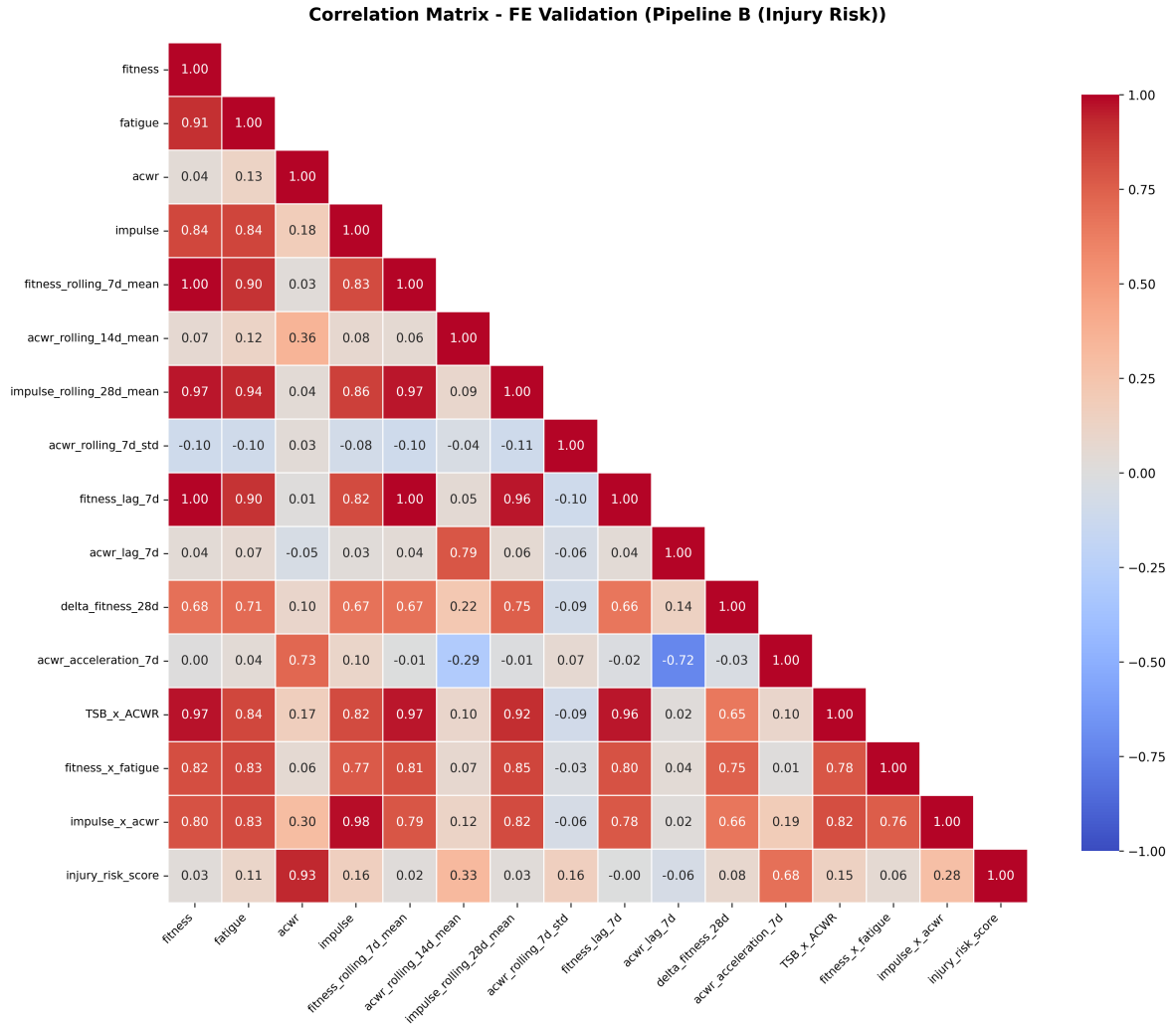

Figura 2: Heatmap correlazioni (Pipeline B): audit FE su subset di feature e target `injury_risk_score`.

Tabella 11: Feature list finale per IMPETUS dopo FE e riduzione (n_features=35).

Categoria	Feature	Descrizione sintetica
Base (Banister)	`impulse`	Stimolo/carico giornaliero sintetico.
Base (Banister)	`tau_F`, `tau_D`	Costanti di decadimento fitness/fatigue (parametri utente).
Base (Banister)	`fitness`, `fatigue`	Stati Banister (fitness e fatigue).
Base (Banister)	`performance`	Performance day-level (scala 0–100) generata dal modello.
Base (Carico)	`acwr`	Acute/Chronic Workload Ratio (solo giorni validi in Pipeline B).
Rolling mean	`acwr_rolling_7d_mean`	Media mobile 7 giorni di ACWR.
Rolling mean	`acwr_rolling_14d_mean`	Media mobile 14 giorni di ACWR.
Rolling mean	`acwr_rolling_28d_mean`	Media mobile 28 giorni di ACWR.
Rolling mean	`fitness_rolling_7d_mean`	Media mobile 7 giorni di fitness.
Rolling mean	`fitness_rolling_14d_mean`	Media mobile 14 giorni di fitness.
Rolling mean	`fitness_rolling_28d_mean`	Media mobile 28 giorni di fitness.
Rolling mean	`impulse_rolling_7d_mean`	Media mobile 7 giorni di impulse.
Rolling mean	`impulse_rolling_14d_mean`	Media mobile 14 giorni di impulse.
Rolling mean	`impulse_rolling_28d_mean`	Media mobile 28 giorni di impulse.
Rolling std	`acwr_rolling_7d_std`	Volatilità ACWR su 7 giorni.
Rolling std	`acwr_rolling_14d_std`	Volatilità ACWR su 14 giorni.
Lag (history)	`acwr_lag_7d`, `acwr_lag_14d`	ACWR a 7/14 giorni prima.
Lag (history)	`fitness_lag_7d`, `fitness_lag_14d`	Fitness a 7/14 giorni prima.
Lag (history)	`fatigue_lag_7d`, `fatigue_lag_14d`	Fatigue a 7/14 giorni prima.
Lag (autoregressivo)	`performance_score_0_10_lag_7d`, `performance_score_0_10_lag_14d`	Lag del target performance (setting autoregressivo).
Lag (autoregressivo)	`injury_risk_score_lag_7d`, `injury_risk_score_lag_14d`	Lag del target risk (setting autoregressivo).
Trend	`delta_fitness_28d`	Differenza fitness rispetto a 28 giorni prima.
Trend	`delta_impulse_28d`	Differenza impulse rispetto a 28 giorni prima.
Trend	`fitness_trend_14d`	Slope media (per giorno) di fitness su 14 giorni.
Trend	`acwr_acceleration_7d`	Accelerazione ACWR (variazione vs 7 giorni prima).
Interaction	`TSB_x_ACWR`	Interazione forma×carico (TSB non usato come feature base).
Interaction	`fitness_x_fatigue`	Interazione tra stati Banister.
Interaction	`impulse_x_acwr`	Interazione volume×ratio (proxy intensità di spike).