Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Magistrale in
Scienze Statistiche

Valutazione dell'efficacia tattica nel tennis ATP

Relatore Prof. Bruno Scarpa
Dipartimento di Scienze Statistiche

Laureando: Filippo Bagarello
Matricola N. 2027163

Anno Accademico 2025/2026

Riassunto

Storicamente, sono stati condotti molti studi che utilizzano metodi statistici per prevedere il risultato finale delle partite di tennis professionistico. Tuttavia, lo scopo di questa tesi è quello di stimare la probabilità marginale condizionata di vittoria a livello di singolo punto in funzione delle decisioni spaziali e tattiche prese da ciascun giocatore.

Il punto di partenza è lo schema di base *serve + 1* e l'uso dei dati di tracciamento sequenziale del *Match Charting Project*. Utilizzando i dati di tracciamento sequenziale, lo studio cerca di estrarre il segnale creato dalle tattiche e di isolarlo dal rumore prodotto da eventi casuali e altre variabili confondenti, tra cui i punti di forza latenti dei giocatori (ancorati attraverso il modello parametrico di Dixon & Coles) e le pressioni psico-fisiche variabili nel tempo applicate ai giocatori.

È stata effettuata un'analisi comparativa tra varie famiglie di modelli di apprendimento statistico in grado di gestire le complesse caratteristiche gerarchiche (i dati sono annidati all'interno delle partite) e temporali dei dati sportivi. Attraverso l'applicazione della validazione *walk-forward*, l'analisi comparativa ha dimostrato che GPBoost era superiore a tutti gli altri modelli concorrenti, grazie alla sua capacità di integrare alberi decisionali *gradient boosting* con effetti casuali, riducendo così il problema della sovraparametrizzazione rispetto sia al tradizionale modello lineare misto che alla rete neurale profonda.

Attraverso tecniche di inferenza predittiva, le complesse probabilità stimate dal modello sono state trasformate in indicazioni tattiche oggettive e direttamente applicabili. I risultati dimostrano una significativa dipendenza geometrica condizionale tra la direzione ottimale del servizio e la traiettoria ottimale dell'attacco. Ciò dimostra il potenziale di un nuovo strumento quantitativo per l'ottimizzazione della strategia e funge anche da base per ulteriori applicazioni nella teoria dei giochi relative alla determinazione degli equilibri per le strategie miste.

Indice

Introduzione

Considerazioni iniziali

Negli ultimi due decenni, il mondo dello sport professionistico ha vissuto una profonda rivoluzione quantitativa. Grazie all'avvento di tecnologie di tracciamento ottico e spaziale ad altissima precisione, il tennis si è trasformato da una disciplina analizzata prevalentemente su basi euristiche e qualitative a un dominio ricco di dati complessi. L'acquisizione di informazioni punto per punto (traiettorie, velocità, rotazioni e posizionamento dei giocatori) ha aperto la strada all'impiego di algoritmi avanzati per l'analisi delle prestazioni.

La via è stata così aperta all'uso di modelli di grande complessità per l'analisi delle prestazioni. Numerosi, tuttavia, sono stati, e sono, i contrasti, in parte strutturali, con cui l'applicazione della statistica e dell'apprendimento automatico in un simile campo si deve necessariamente scontrare. Buona parte del materiale presentato in letteratura ha riguardato la previsione del risultato finale dell'incontro. Questo approccio a larga scala, per quanto abbia raggiunto quote predittive confortanti (molte volte superiori al 70%), beneficia della legge dei grandi numeri: aggregando decine o centinaia di "punti", la varianza intrinseca del gioco si riduce, rendendo il risultato del match fortemente dipendente dalla probabilità media di vittoria al servizio dei due giocatori (modelli stocastici e catene di Markov).

La previsione del singolo punto

Quando si passa dal macro-evento dell'incontro al micro-evento dello scambio, l'analisi tennistica si scontra con un ostacolo importante. La previsione dell'esito di un punto, basandosi unicamente sui primi colpi, è difficile per via dell'alto tasso di incertezza aleatoria. Elementi accidentali come un nastro, una deviazione minima, un soffio di vento infondono un rumore statistico che rende la previsione del singolo punto ad alta stocasticità.

A fronte di questa difficoltà, gli approcci statistici tradizionali, come i modelli parametrici, tendono ad appiattire la previsione sulle abilità storiche medie dei giocatori, fallendo nel catturare le sfumature tattiche che si innescano istante per istante sul campo da gioco.

Obiettivi e contributo dell'elaborato

Lo scopo principale di questo lavoro di tesi non consiste nella sola massimizzazione dell'accuratezza predittiva dei modelli proposti, bensì nel fornire uno strumento da cui estrarre regole tattiche spiegate a partire dai dati passati.

Al fine di separare in maniera netta l'informazione tattica rispetto al rumore di fondo, si è stabilito che l'indagine deve limitarsi a studiare esclusivamente la dinamica di avvio dello scambio, valutando i dati sulla base delle sole sequenze ad almeno tre colpi, il cosiddetto schema *Serve + 1* (servizio, risposta e terzo colpo).

Sfruttando le caratteristiche geometriche di questi tre colpi, ci si pone come obiettivi quanto segue:

- Illustrare come l'uso di algoritmi che si fondano sul partizionamento ricorsivo, come gli alberi decisionali e il *Gradient Boosting*, possa rivelarsi più efficace rispetto ai modelli parametrici tradizionali nella decifrazione di stili di gioco caratterizzati da una significativa asimmetria. Per questo scopo, sono stati scelti come soggetti di studio i tennisti Matteo Berrettini e Stefanos Tsitsipas, il cui marcato contrasto di affidabilità tra il colpo di dritto e quello di rovescio rappresenta un esempio ideale per l'estrazione di regole logiche non lineari.

- Affrontare l'alta variabilità del tennis integrando modelli misti muniti di effetti casuali, mediante il *GPBoost*, che sono capaci di gestire le fluttuazioni delle prestazioni del tennista all'interno di ciascun incontro.

- Convertire le previsioni *black box* generate dagli algoritmi in strumenti di supporto decisionale, con l'obiettivo di creare una matrice di simulazione tattica. Quest'ultimo strumento può essere di grande utilità per lo staff tecnico, in quanto consente di identificare le vulnerabilità dell'avversario, di concentrare le sessioni di allenamento e di massimizzare l'imprevedibilità delle scelte geometriche effettuate in campo.

Struttura della tesi

L'attuale elaborato si articola in sei capitoli principali, ordinati in modo da costruire un percorso di lettura che va dalla teoria di base all'applicazione pratica in campo.

Capitolo 1 – Stato dell'arte e revisione della letteratura: inquadra il problema predittivo nel tennis, analizzando come l'incremento dei dati disponibili abbia esteso i limiti dei modelli tradizionali e descrivendo il modo che in letteratura si è sino ad ora usato per decifrare le prospettive di gioco.

Capitolo 2 – Estrazione e pulizia della base di dati: descrive le fonti dei dati e approfondisce le fasi di bonifica delle stringhe alfanumeriche che descrivono gli scambi,

del riempimento dei buchi informativi e dell'applicazione del filtro a selezione dei primi tre colpi dell'azione di gioco.

Capitolo 3 – Ingegneria delle variabili e contestualizzazione: tratta particolarmente della costruzione del predittore spaziale e tattico, con un cenno particolare alla stima della base storica dei giocatori secondo lo schema di regressione penalizzata (ridge).

Capitolo 4 – Metodologie e modelli predittivi: È rappresentato dall'impianto algoritmico della tesi. Si analizzano il compromesso tra segnale e rumore, la convalida di tipo temporale sequenziale e il funzionamento logico-matematico dei modelli confrontati (lineari, ad albero, d'insieme, *deep learning*).

Capitolo 5 – Risultati e analisi delle prestazioni: contiene il confronto delle metriche di valutazione, dimostrando l'esito favorevole degli algoritmi ad albero, discute il confronto tra modelli additivi, con illustrazione visiva delle tattiche adoperate.

Capitolo 6 – Simulazione tattica e applicazioni pratiche: applica i risultati ad uno scenario reale, proponendo l'utilizzo della matrice di simulazione per la preparazione dei match, lo studio dell'avversario (e l'applicazione della teoria dei giochi per ridurre la prevedibilità in campo).

Capitolo 7 – Conclusioni: considerazioni finali sui risultati ottenuti, limiti dei metodi proposti e possibili sviluppi futuri.

Capitolo 1

Stato dell'arte e revisione della letteratura

Negli ultimi dieci anni la modellazione statistica del tennis ha conosciuto una fortuna crescente grazie all'introduzione di dispositivi quali l'Hawk-Eye , sofisticato sistema di tracciamento ottico multi-camera impiegato in tornei professionistici per la ricostruzione tridimensionale e millimetrica delle traiettorie della pallina, mediante il quale il campo di gioco è diventato il generatore di una grande mole di dati spazio-temporali a densità elevatissima. Come rilevato da recenti articoli scientifici di ambito tennistico in riviste di *machine learning* (Vives et al. 2024), l'opera di ricerca si è via via orientata da semplici analisi descrittive a processi di *Knowledge Discovery in Databases* (KDD) volti a stabilirne pattern tattici non banali.

1.1 La casualità nel tennis: l'incontro vs. il singolo punto

La letteratura scientifica si è spesso concentrata sulla previsione dell'esito dell'incontro. Gli studi di Sipko e Knottenbelt (2015) evidenziano una promettente modalità di utilizzo di regressione logistica, *machine learning* supervisionato e reti neurali artificiali, dimostrando di poter superare le prestazioni dei tradizionali modelli stocastici (basati sulle catene di Markov), con accuratezza predittiva prossima al 70%.

Tuttavia, la previsione di un intero incontro beneficia della legge dei grandi numeri: aggregando centinaia di eventi (i "punti"), la varianza stocastica tende a compensarsi, permettendo alla reale differenza di valore tra i giocatori di emergere. Al contrario, quando l'obiettivo si restringe alla previsione del singolo punto, la complessità aumenta di molto. Il recente lavoro di Hovad, Mikkelsen, e Illum (2025) evidenzia la forte componente aleatoria nell'esito del singolo punto nel tennis. Gli autori confrontano diversi modelli di machine learning — in particolare Extreme Gradient Boosting (Chen e Guestrin 2016) e Random Forest — con un modello di riferimento costruito come frequenza storica di vittoria del

giocatore al servizio. I risultati mostrano che i modelli avanzati riescono a migliorare solo marginalmente tale riferimento empirico, suggerendo che una quota rilevante della variabilità dell'esito del punto non è spiegabile con le variabili da loro disponibili.

1.2 Analisi del processo dello scambio

Per superare i limiti delle statistiche aggregate, Makino et al. (2020) analizzano la struttura dello scambio a livello di sequenze di colpi. Utilizzando la selezione LASSO, gli autori mostrano che i pattern tattici intra-punto (microstrategia) forniscono capacità predittiva apprezzabile (AUC ≈ 0.69), suggerendo che la dinamica dello scambio è più informativa rispetto a statistiche riassuntive a livello di incontro, quali la percentuale di punti vinti al servizio o in risposta.

In parallelo, grazie ai dati di tracciamento forniti da Hawk-Eye, Wei et al. (2016) propongono di definire i dizionari di stile. Il loro approccio si propone di modellare l'interazione specifica tra i giocatori, dimostrando che il posizionamento spaziale e la tipologia del colpo precedente hanno un forte impatto sulla probabilità di vincita del colpo successivo. Questa spiegazione, se esatta, vorrebbe dire che il gioco del tennis non è un processo a stati indipendenti, bensì una sequenza di stati condizionali dove lo stato a cui si è giunti (la posizione, il tipo di colpo) determina lo spazio delle probabilità.

1.3 Non-stazionarietà ed importanza del punteggio

Un contributo di assoluta importanza per spiegare la variabilità nel tennis viene dallo studio della *clutch performance*, ovvero della prestazione sotto pressione. Kovalchik e Reid (2016) hanno introdotto il concetto di *clutch averaging*, ovvero quello di pesatura delle statistiche della prestazione sulla base dell'importanza relativa di ciascun punto, dal momento che la prestazione del giocatore non è la stessa in tutti gli istanti, essendovi punti che, per la loro importanza, debbono essere considerati di gran lunga superiori agli altri e rivestono così una condizione di non-stazionarietà. Ad esempio, un punto giocatore su palla break a sfavore in un game decisivo per il set è porta un giocatore a colpire e prendere decisioni tattiche con una pressione decisamente maggiore rispetto ad un punto giocato sullo 0-0 a inizio partita. Tale aspetto appare cruciale in sede di modellazione: il modello deve poter distinguere se un pattern tattico sia condizionato dalla situazione di punteggio, poiché i tennisti variano la loro spregiudicatezza nei momenti cruciali dell'incontro.

1.4 Posizionamento della ricerca e sintesi statistica

Dalla revisione della letteratura emerge una netta dicotomia: da un lato modelli macroscopici accurati ma privi di dettagli tattici, dall'altro modelli spazio-temporali estremamente precisi

ma spesso basati su dati proprietari e non scalabili.

La presente tesi si posiziona in modo originale in questo scenario, adottando un approccio di filtraggio tattico mirato: l'isolamento della sequenza a tre colpi.

Dal punto di vista statistico, questo lavoro mira a ridurre la dimensionalità del problema, concentrando la capacità predittiva dei modelli su un sottoinsieme di eventi (il *Serve + 1*) dove il rumore aleatorio è parzialmente mitigato dalla dominanza del giocatore al servizio. Tale scelta verrà argomentata in modo più approfondito in seguito.

L'integrazione di modelli gerarchici con modelli di tipo boosting, ovvero GPBoost (Sigrist 2022), ci consente di trattare l'incontro come effetto casuale tramite la stima della varianza inter-match. In questo modo viene isolata la componente tattica stabile (lo stile) da componenti esterne e specifiche di un determinato incontro, set o game quali forma fisica dei giocatori che oscilla nel tempo, condizioni ambientali o climatiche. Di conseguenza, i risultati dei modelli non serviranno solo a convalidare i modelli stessi, ma aiuteranno anche a scoprire le regole di decisione dei giocatori che spiegano, ad esempio, come Stefanos Tsitsipas risulti dominante in precisi assetti geometrici dello scambio.

Capitolo 2

Estrazione e Pulizia della Base di Dati

Nel contesto del tennis professionistico, la raccolta di dati precisi per ogni singolo colpo risulta complessa. I dati, infatti, non vengono resi pubblici e spesso restano di proprietà delle aziende partner del circuito incaricate della raccolta. Di conseguenza è difficile reperire dati di qualità. In questo capitolo si presenta la fonte dei dati usati nella tesi, le caratteristiche delle variabili e i tanti controlli e pulizie effettuati per assicurarsi che i dati siano di buona qualità.

2.1 Il *Match Charting Project* e la selezione del campione

Un ruolo centrale in questo lavoro è svolto dal *Match Charting Project* (MCP) (Sackmann 2025), un'iniziativa collaborativa promossa dall'analista sportivo Jeff Sackmann. Lanciato ufficialmente alla fine del 2013, il progetto si è rapidamente espanso fino a costituire oggi il più grande database *open-source* di dati colpo su colpo (*shot-by-shot*) disponibile pubblicamente per il mondo del tennis. L'archivio ospita attualmente la mappatura di oltre 10.000 incontri professionistici, coprendo in modo estensivo i circuiti maggiori maschili (ATP) e femminili (WTA), includendo persino la catalogazione di decine di partite storiche risalenti ai decenni passati.

Il funzionamento operativo del MCP si basa su una rigorosa e complessa sintassi di codifica testuale. Una vasta comunità di appassionati volontari (chiamati *charters*) analizza le registrazioni video degli incontri e trascrive l'intera sequenza di gioco di ogni singolo punto utilizzando una specifica notazione stenografica. Questa grammatica standardizzata permette di tracciare con precisione la direzione del servizio, la tipologia di colpo (dritto, rovescio, *slice*, volée, *smash*) e la traiettoria spaziale e di profondità di ogni impatto. In questo modo, l'evoluzione dinamica e tattica di un intero scambio viene letteralmente

convertita in una stringa alfanumerica strutturata, rendendo l'informazione processabile da algoritmi informatici.

Nonostante gli evidenti limiti legati alla natura del *crowdsourcing* (come potenziali errori di digitazione o leggere differenze di giudizio soggettivo tra i vari valutatori), il valore aggiunto del MCP risiede proprio in questa meticolosa ricostruzione manuale delle sequenze. Questo approccio permette di raccogliere dati dettagliati, raggiungendo una profondità esplorativa quasi confrontabile a quella dei costosi sistemi di tracciamento ottico automatici (*Hawk-Eye*). Inoltre, tale metodologia di raccolta integra intrinsecamente l'interpretazione umana nella classificazione dei colpi, consentendo di rilevare dettagli strategici e sfumature tattiche che molto spesso sfuggono ai processi puramente automatizzati basati sulla sola telemetria.

Si è lavorato con dati a livello di singoli "punti" (ogni riga è un "punto"), estraendo tutti gli incontri ATP (maschili) svolti tra il 1 gennaio 2020 e il 14 giugno 2025. La fine dell'intervallo corrisponde alla data di ultimo aggiornamento disponibile del dataset al momento di avvio delle analisi. L'inizio dell'intervallo temporale è stato fissato al 2020 poiché includere i dati delle stagioni precedenti comporterebbe una distorsione dovuta all'evoluzione delle attrezzature tecnologiche (come le racchette e i materiali delle corde) e ai cambiamenti biomeccanici e stilistici che sono avvenuti nel tennis contemporaneo. Il campione selezionato riflette, pertanto, lo stato attuale del gioco moderno. Alcune variabili di contesto (fisse per l'intero incontro) sono state aggregate al dataset dei punti estraendo le informazioni dal corrispondente dataset a livello di *match* (ogni riga è una partita), consultabile tra il materiale messo a disposizione dal MCP.

2.2 Struttura gerarchica e codifica delle variabili

Come anticipato in precedenza, i dati grezzi sono organizzati in una struttura relazionale composta da due archivi principali: uno dedicato ai metadati a livello di incontro (date, superfici, tornei) e uno relativo alla scomposizione dettagliata dei punti. La base dati adotta un'architettura gerarchica annidata, in cui l'unità statistica fondamentale (il "punto") è inclusa in un gioco (*game*), che a sua volta appartiene a un *set*, l'insieme dei quali costituisce l'incontro (*match*).

Le variabili più rilevanti, estratte per la costruzione dei modelli predittivi, si dividono in tre categorie: contestuali, sequenziali e di esito.

2.2.1 Variabili di contesto

Queste variabili descrivono lo stato del sistema prima dell'inizio dello scambio.

- `match_id`: identificativo univoco dell'incontro (es. *20250608-M-Roland_ Garros-F-Jannik_ Sinner-Carlos_ Alcaraz*).

- `Surface:` tipo di superficie di gioco (*cemento, terra rossa, erba*).
- `Pt, Set1, Set2, Pts:` progressione numerica del punto e dello stato del punteggio al momento esatto in cui la pallina viene messa in gioco.
- `Svr:` variabile dicotomica che indica quale dei due giocatori è al servizio.

2.2.2 Sintassi dello scambio

Il dataset è costituito dalle variabili `X1st` e `X2nd`, che contengono una stringa codificata che descrive l'intera sequenza dello scambio, rispettivamente dopo la prima o la seconda palla di servizio. La sintassi MCP segue regole posizionali precise. Ogni colpo è rappresentato da una combinazione alfanumerica che indica, in sequenza: la direzione del servizio, il tipo di esecuzione dei colpi successivi, la direzione e il tipo di esito finale. Ad esempio, la direzione del servizio è codificata numericamente: 4 (ad uscire), 5 (al corpo), 6 (alla T). I colpi successivi al servizio specificano la natura dell'esecuzione (f per dritto/forehand, b per rovescio/backhand) e la direzione nel campo avversario (1 per incrociato, 2 per centrale, 3 per lungolinea). La concatenazione di questi caratteri consente di ricostruire con precisione la geometria e la dinamica del punto.

2.2.3 Esito del punto (terminatori)

I caratteri terminali delle stringhe codificano la conclusione dello scambio:

- ***** : Vincente
- **@** : Errore non forzato
- **#** : Errore forzato

La variabile risposta `PtWinner` identifica il giocatore che ha ottenuto il punto (1 per il giocatore 1, 2 per il giocatore 2).

2.3 Pulizia e qualità dei dati

Prima di procedere alla modellazione, la base di dati ha richiesto un massiccio intervento di pulizia, principalmente di regolarizzazione sintattica, indispensabile per rendere le stringhe comprensibili dagli algoritmi.

Sono state individuate e corrette alcune anomalie nei dati, ad esempio l'incontro Griekspoor-Cobolli, Davis Cup 2024 che risultava diviso e mischiato su più righe, omissioni di indicatori relativi ai tie-break, dati duplicati e molti altri casi. Successivamente, si è fatto ricorso a un'analisi sintattica basata su espressioni regolari, che ha permesso di correggere modificatori testuali anomali e risolvere ambiguità relative alla direzione dei colpi (ad

esempio, la presenza di più cifre consecutive non riconducibili a un singolo colpo). È stata inoltre adottata una procedura di imputazione: laddove risultasse assente l'informazione sulla direzione del colpo (in oltre 14.000 casi), è stata attribuita la classe generica "0", mentre nei casi di mancata indicazione sul tipo di colpo si è imputato con la generica classe "q", così da mantenere la coerenza della struttura dello scambio senza eliminare dati potenzialmente informativi. Entrambe le classi generiche sono previste nella sintassi MCP.

Il dettaglio completo delle operazioni di pulizia ai dati è consultabile infondo in appendice A.

2.4 Analisi descrittiva pre filtro

2.4.1 Lunghezza degli scambi

La distribuzione del numero di colpi per punto costituisce la giustificazione empirica alla base dell'intera tesi. La maggior parte dei punti nel tennis professionistico moderno si conclude entro i primi colpi. L'analisi di frequenza della lunghezza degli scambi, misurata tramite la lunghezza della stringa MCP, mostra un decadimento asimmetrico e marcatamente asintotico.

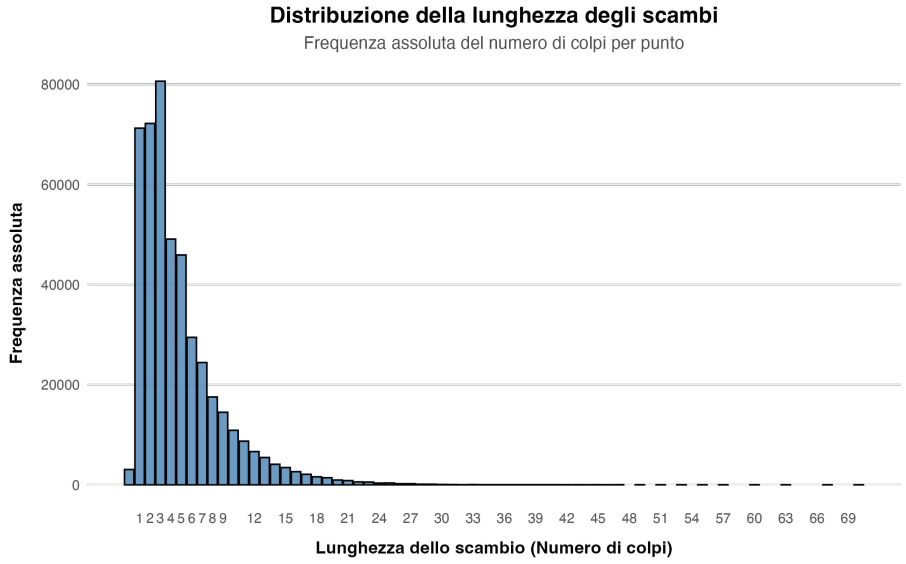


Figura 2.1: Grafico a barre della lunghezza degli scambi. La marcata asimmetria positiva (coda a destra) evidenzia che i punti brevi prevalgono nel gioco contemporaneo, il che giustifica il focus analitico sui primi tre colpi.

Come evidenziato dall'istogramma in Figura 2.1, lo schema concettuale noto come *Serve + 1* (che include il servizio, la risposta e il terzo colpo) rappresenta la porzione statisticamente più rilevante della prestazione di un tennista, costituendo la base per il filtraggio avanzato descritto nel capitolo successivo.

Capitolo 3

Ingegneria delle variabili e contestualizzazione

Nel processo di esplorazione dei dati, la fase di ingegneria delle variabili assume un ruolo fondamentale. Questo capitolo si concentra sulla trasformazione dei dati grezzi in un insieme di predittori matematici capaci di rappresentare le dinamiche complesse di un incontro di tennis. L'attività si è sviluppata attraverso la costruzione di indicatori contestuali, metriche psico-fisiche non stazionarie e, in particolare, la stima di una probabilità storica di base mediante modelli di regressione regolarizzata.

3.1 Costruzione dello spazio di stato: variabili contestuali

Ogni punto nel tennis rappresenta un evento fortemente influenzato dallo "spazio degli stati" in cui si verifica. Al fine di istruire in modo efficace gli algoritmi di apprendimento automatico, sono state identificate e ingegnerizzate le seguenti dimensioni contestuali:

- **Superficie di gioco (`Surface`):** codificata come variabile categorica (cemento, terra rossa, erba). La superficie altera la cinematica della pallina e influenza l'efficacia dei colpi (es. il rimbalzo alto sulla terra battuta favorisce l'uso delle rotazioni).
- **Importanza del Torneo (`tournament_importance`):** Per quantificare la pressione agonistica, è stata creata una variabile ordinale basata sui punti assegnati dal circuito ATP: i Tornei del Grande Slam (2000 punti), i Masters 1000 (1000 punti), gli ATP 500, gli ATP 250 e i circuiti minori.
- **Vantaggio e punteggio relazionale:** Invece di utilizzare il punteggio assoluto (ad esempio, 40-30), è stata sviluppata una variabile di vantaggio differenziale che rappresenta la distanza del servitore dalla vittoria del gioco (ad esempio, $+1$ in caso di vantaggio, -1 in caso di svantaggio). Tale approccio consente di ridurre la dimensionalità e di migliorare la generalizzazione del modello.

3.2 Dinamiche non stazionarie: affaticamento, inerzia e pressione

Kovalchik e Reid (2016) evidenziano che la prestazione umana nello sport è soggetta a fluttuazioni. Sono state pertanto sintetizzate tre misure per quantificare lo stato psico-fisico latente dei giocatori in ogni istante:

- **Affaticamento e Inerzia Psicologica:** L'affaticamento (*fatigue*) è stato stimato in 3 metriche calcolando il volume di gioco cumulato, cioè il numero totale di colpi giocati fino a un certo momento, sia dal giocatore 1 che dal giocatore 2. Nello specifico, si calcola la fatica comulata come somma del numero di colpi giocati dall'inizio dell'incontro, la fatica recente come somma del numero di colpi giocati negli ultimi 15 punti, la fatica immediata come numeri di colpi giocati nel punto precedente. Disponendo di dati spaziali *Hawk-Eye*, sarebbe più accurato stimare la stanchezza calcolando i metri persorsi da ogni atleta nel corso della partita. L'inerzia psicologica (*momentum*) è stata modellata usando medie mobili sui punti più recenti. In particolare, è stata calcolata la percentuale di punti vinti negli ultimi 15 e 10 giocati, per permettere al modello di rilevare eventuali picchi di forma o cali temporanei di concentrazione.
- **Importanza del punto:** Seguendo il paradigma di Kovalchik e Reid (2016), non tutti i punti influenzano allo stesso modo l'esito del set. È stata quindi creata una variabile che indica la pressione, distinguendo tra punti di routine (come 15-0) e punti critici o di rottura (palle *break*, *set point*). Questa variabile aiuta l'algoritmo a valutare se i giocatori cambiano le scelte direzionali quando la pressione competitiva è alta (*importanza*).

3.3 Scomposizione spaziale dei colpi

Dal punto di vista algoritmico, la parte più complessa è stata dividere le stringhe testuali ripulite in vettori posizionali. Per ogni punto, la sequenza è stata suddivisa per isolare le caratteristiche esatte dei primi quattro impatti con la pallina:

1. **colpo 1 (Servizio):** si è estratta solo la direzione ("Ad uscire", "Al corpo", "Alla T").
2. **colpo 2 (risposta):** estrazione della tipologia (dritto, rovescio, *slice*, ecc.), della direzione (*Cross*, centrale, lungolinea) e della profondità stimata (corta, media, profonda).
3. **Colpo 3 (terzo colpo o *serve+1*):** Estrazione della tipologia (dritto o rovescio) e della direzione geometrica scelta dal servitore in uscita dal movimento di battuta.

4. **Colpo 4:** Estrazione della reazione del risponditore (tipo e direzione).

3.4 Modellazione della forza latente: probabilità storica di base

Un problema intrinseco nella previsione degli esiti nel tennis è rappresentato dall'elevato rischio di sovradattamento (*overfitting*). Se si stima che una specifica combinazione tattica ha avuto successo, potrebbe erroneamente attribuire il merito al colpo stesso, trascurando che l'esito positivo era in realtà determinato da un significativo divario di forza tra i due tennisti.

Per fornire ai modelli una linea di base solida, è stato necessario stimare a priori la probabilità che il giocatore al servizio vincerà il punto, in funzione della forza storica del suo avversario. Ispirandosi all'approccio di Dixon e Coles (1997) alla modellazione delle abilità difensive e offensive nel gioco del calcio, è stato formulato un modello, riadattato alla struttura microscopica del tennis.

Nello specifico, la probabilità p_{ij} che il giocatore i (al servizio) vinca il punto contro il giocatore j (in risposta) è stata parametrizzata come:

$$\text{logit}(p_{ij}) = \mu + \alpha_i - \beta_j \tag{3.1}$$

Dove μ rappresenta il parametro di intercetta globale (il vantaggio intrinseco e medio del servizio nel circuito professionistico), α_i quantifica l'abilità latente offensiva (al servizio) del giocatore i, e β_j misura l'abilità latente difensiva (in risposta) del giocatore j.

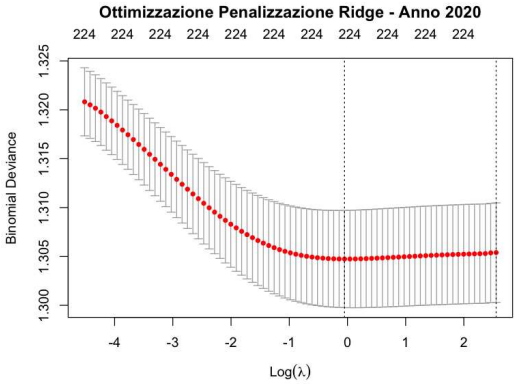
Tuttavia, inserire gli identificativi dei giocatori come semplici variabili indicatrici in una regressione logistica standard su un circuito con centinaia di tennisti avrebbe comportato problemi di multicollinearità e un'esplosione della varianza per i giocatori con un numero ridotto di osservazioni (es. giovani esordienti o *wild card*).

In questo studio è stata usata una regressione logistica con penalizzazione Ridge (penalizzazione L_2) (Hoerl e Kennard 1970), ottimizzata con il pacchetto `glmnet`. La funzione di costo minimizzata è la seguente:

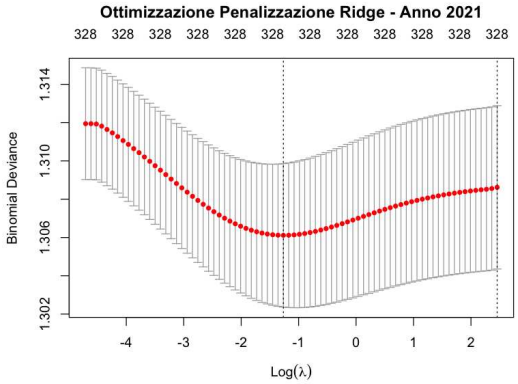
$$\mathcal{L}(\mu, \alpha, \beta) = -\sum_{k=1}^{n} [y_k \log(p_k) + (1 - y_k) \log(1 - p_k)] + \lambda \left(\sum_i \alpha_i^2 + \sum_j \beta_j^2 \right) \tag{3.2}$$

Qui, $y_k \in \{0, 1\}$ indica l'esito del punto k-esimo, p_k è la probabilità stimata per quel punto e λ è l'iperparametro di regolarizzazione, calibrato con la convalida incrociata a 10 gruppi (*10-fold cross-validation*). Viene selezionato il λ che minimizza l'errore di previsione, come osservabile nei pannelli in Figura 3.1. Questo metodo ha ridotto i coefficienti α e β dei

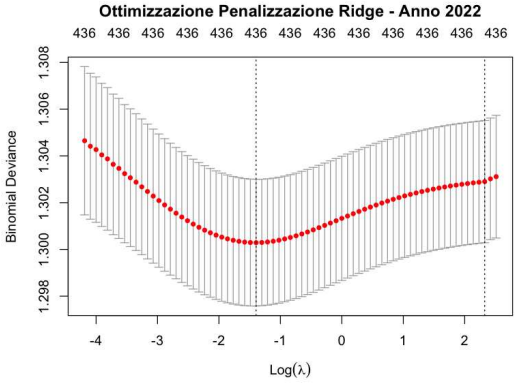
tennisti meno noti verso lo zero, cioè verso la media globale del circuito, eliminando la varianza dovuta a pochi dati.



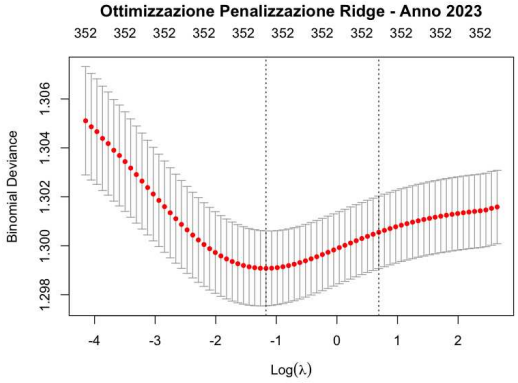
(a) Stagione 2020



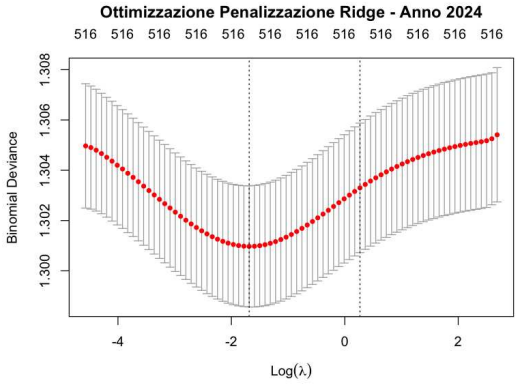
(b) Stagione 2021



(c) Stagione 2022



(d) Stagione 2023



(e) Stagione 2024

Figura 3.1: Ottimizzazione del modello Ridge nei diversi periodi analizzati. I grafici mostrano l'andamento della devianza binomiale al variare del parametro di penalizzazione λ. La prima linea verticale indica il λ ottimale scelto tramite convalida incrociata.

Per evitare di usare informazioni future nella previsione di eventi passati (*data leakage*) e per seguire l'evoluzione reale della forma fisica, l'adattamento è stato svolto in modo iterativo su base stagionale. I parametri μ, α_i e β_j sono stati stimati per ogni giocatore in ogni anno solare. Per ogni punto giocato nell'anno t, i coefficienti usati per calcolare la probabilità sono stati stimati solo sui dati dell'anno precedente $(t-1)$.

Dopo aver ottenuto i parametri storici penalizzati, la probabilità di base è stata calcolata usando la trasformazione logistica inversa (funzione *expit*):

$$p_{ij} = \frac{\exp(\mu + \alpha_i - \beta_j)}{1 + \exp(\mu + \alpha_i - \beta_j)} \tag{3.3}$$

Il valore risultante è stato infine integrato nel dataset come variabile numerica continua (chiamata `Player_baseline_prob`). Questa metrica, calcolata sempre su dati *out-of-sample*, sarà un importante punto di riferimento per costruire tutti gli algoritmi successivi.

3.4.1 Profilazione dinamica dell'avversario: metriche a finestra mobile

Ad integrazione della probabilità storica di base (`Player_baseline_prob`) — la quale assorbe il divario tecnico e fisico strutturale tra i due tennisti basandosi sui dati dell'intera stagione precedente — è stato deciso di sviluppare un insieme di predittori volti a catturare lo stato di forma e le tendenze di gioco più recenti. Mentre la baseline storica fornisce un posizionamento di lungo periodo, queste nuove variabili catturano l'informazione dinamica dell'ultimo anno di gioco.

A partire dai dati riepilogativi forniti dal file *charting-m-stats-Overview.csv*, sono state ingegnerizzate quattro metriche di prestazione calcolate su una finestra temporale mobile di 365 giorni antecedenti alla data del match. Il giorno stesso dell'incontro è stato rigorosamente escluso dal calcolo per prevenire qualsiasi rischio di *data leakage* (utilizzo di informazioni future).

Le variabili ingegnerizzate sono le seguenti:

- Efficacia al servizio: la capacità complessiva di dominare i propri turni di battuta.

$$\texttt{Serve Effectiveness} = \frac{\texttt{First Won} + \texttt{Second Won}}{\texttt{Serve Pts}}$$

- Qualità in risposta: la percentuale di punti vinti durante i turni di battuta avversari.

$$\texttt{Return Quality} = \frac{\texttt{Return Pts Won}}{\texttt{Return Pts}}$$

- Tasso di Ace: l'incidenza dei servizi vincenti diretti, fondamentale indicatore di potenza e precisione pura.

$$\texttt{Ace Rate} = \frac{\texttt{Aces}}{\texttt{Serve Pts}}$$

- Indice di Aggressività: la propensione a prendere l'iniziativa e chiudere il punto, accettando il rischio fisiologico dell'errore.

$$\texttt{Aggression Score} = \frac{\texttt{Winners} + \texttt{Unforced Errors}}{\texttt{Serve Pts} + \texttt{Return Pts}}$$

Dal punto di vista statistico, per il calcolo di tali indici si è optato per un approccio di aggregazione micro-average (micro-media). Invece di calcolare la media delle percentuali ottenute in ogni singolo match passato, sono stati sommati gli eventi assoluti (es. il totale degli *ace* in un anno) e divisi per il denominatore cumulato (es. il totale dei servizi giocati nell'anno).

Questa scelta metodologica è cruciale per minimizzare la varianza introdotta da partite anomale, eccessivamente brevi o interrotte per ritiro. Tali eventi tendono infatti a generare statistiche estreme (*outliers*) che distorcerebbero una media classica. L'approccio *micro-average* garantisce che le metriche riflettano la reale consistenza stocastica del giocatore su ogni singolo punto disputato, pesando implicitamente di più gli incontri lunghi in cui la vera caratura tecnica dell'atleta ha il tempo di emergere.

In fase di costruzione del database, le metriche sono state originariamente calcolate per entrambi i giocatori in campo (con i prefissi `p1_` e `p2_`). Tuttavia, conformemente al filtro che restringe il dominio all'analisi di uno specifico giocatore di riferimento, in fase di esportazione sono state mantenute esclusivamente le variabili relative all'avversario:

- `opp_serve_eff`
- `opp_return_qual`
- `opp_ace_rate`
- `opp_aggression`

Questa architettura informativa permette ai modelli di contestualizzare le scelte geometriche del giocatore studiato non solo in base alla situazione di gioco in campo, ma anche in reazione allo specifico stato di forma e al livello di aggressività del tennista che si trova oltre la rete in quel preciso periodo storico.

3.5 Il filtro tattico: servizio e terzo colpo

Dopo aver aggiunto i predittori storici e fisici al dataset, è stato applicato un filtro concettuale per definire il campo di studio della tesi. Lo scopo della ricerca non è prevedere in generale chi vincerà un punto, ma valutare quanto siano efficaci le scelte tattiche di direzione all'inizio dello scambio.

Per questo motivo, il database è stato filtrato in modo rigoroso, mantenendo solo le righe che arrivano almeno al terzo colpo. Sono stati eliminati:

- Gli *ace* e i doppi falli (punti conclusi al primo colpo).
- I servizi vincenti senza risposta, gli errori in risposta, le risposte vincenti dell'avversario (punti conclusi al secondo colpo).

Questo isolamento dello schema, chiamato **Servizio e Terzo Colpo** (*Serve + 1*), cambia il problema dal punto di vista matematico. Ora la probabilità stimata non è più quella marginale di vincere il punto al servizio, ma quella condizionata di vittoria, dato che la palla è entrata in gioco e lo scambio è iniziato. Eliminando il rumore dei punti brevi, l'algoritmo si concentra sulla relazione tra la geometria dei colpi e il risultato finale.

3.6 Filtro per entità di studio (isolamento del giocatore)

L'ultima fase dell'ingegneria dei dati serve a focalizzare il modello. Adattare un modello decisionale su tutto il circuito ATP porterebbe a un modello ibrido, in cui le regole tattiche di giocatori difensivi e attaccanti si mescolano, producendo una media poco utile. Con gli stessi dati (pre filtro) e gli stessi modelli utilizzati in questo studio e presentati in seguito, sarebbe possibile svolgere analisi analoghe su specifiche dinamiche di gioco differenti, come ad esempio sulle direzioni del servizo o sulle risposte.

Il database finale è stato quindi suddiviso per selezionare solo i punti giocati da un tennista specifico, sia in risposta che al servizio, che sarà l'entità di studio. Nelle prossime sezioni saranno presentati i risultati ottenuti applicando questa struttura ai profili di giocatori con una forte asimmetria tattica, come Stefanos Tsitsipas, per cui il terzo colpo è lo strumento principale di dominanza territoriale.

3.7 Analisi esplorativa: il caso di Stefanos Tsitsipas

Prima dell'applicazione dei modelli e delle successive simulazioni, è necessario eseguire un'analisi esplorativa dei dati per illustrare le caratteristiche tattiche iniziali del giocatore studiato. In questa sezione, analizziamo le frequenze relative delle scelte spaziali effettuate da Stefanos Tsitsipas durante i suoi giochi di servizio per fornire una descrizione della sua strategia tattica e delle sue prestazioni.

In primo luogo, è stata studiata la distribuzione delle direzioni di servizio. Dal grafico a barre che illustra le direzioni di servizio, possiamo vedere che c'è un'evidente distinzione tattica. Tsitsipas utilizza i primi servizi per saggiare i limiti del campo posteriore dell'avversario (pertanto quasi il 50% dei primi servizi sono "larghi" o "lungo la T", ovvero rispettivamente il 46,4% e il 45,1%) e tentare di conquistare il punto con un *ace* o un servizio vincente. I secondi servizi mostrano un cambiamento drastico nella traiettoria (spesso assumendo la forma di servizi conservativi o ricchi di *spin* o *kick*). La maggior parte dei secondi servizi è diretta al corpo dell'avversario (41,3%). Dall'altra parte, le

direzioni dei servizi larghi (28,7%) e lungo la linea centrale (30,0%) registrano una marcata diminuzione. Pertanto, Tsitsipas cerca di ridurre il rischio di doppio fallo, impedendo al contempo al ribattitore di essere eccessivamente aggressivo.

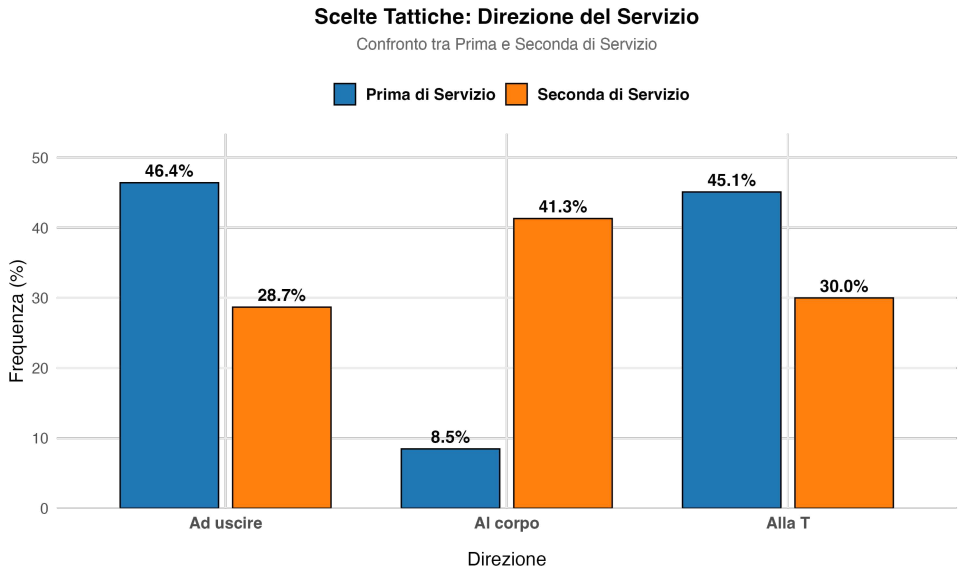


Figura 3.2: Percentuale delle direzioni della prima e della seconda di servizio per Stefanos Tsitsipas.

La differenziazione tattica di Tsitsipas si riflette nelle sue metriche di successo. Ad esempio, come illustrato dall'analisi dei punti vinti (Figura 3.3), Tsitsipas converte il 61,3% di tutti i punti giocati sul primo servizio (ovvero 2.555 punti su 4.168) per vincere il punto. Al contrario, sul secondo servizio, il tasso di conversione di Tsitsipas è del 51,4% (1.521 punti su 2.960).

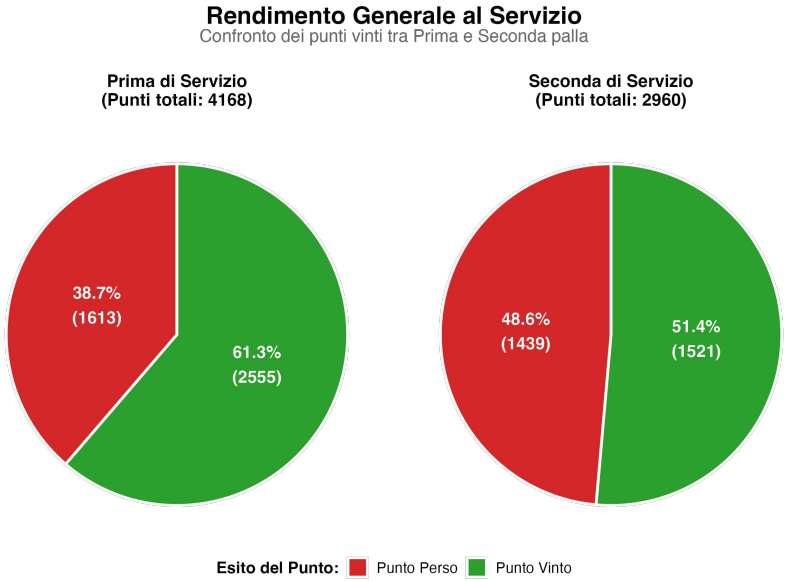


Figura 3.3: Rendimento generale al servizio: confronto dei punti vinti tra prima e seconda palla per Stefanos Tsitsipas.

Si tracciano le curve di densità della percentuale di punti vinti al servizio calcolate sull'intero campione di giocatori ATP disponibile nel dataset. Come mostrato nella Figura 3.4, il numero mediano di punti vinti al servizio nel *tour*[1] è del 62,7%.

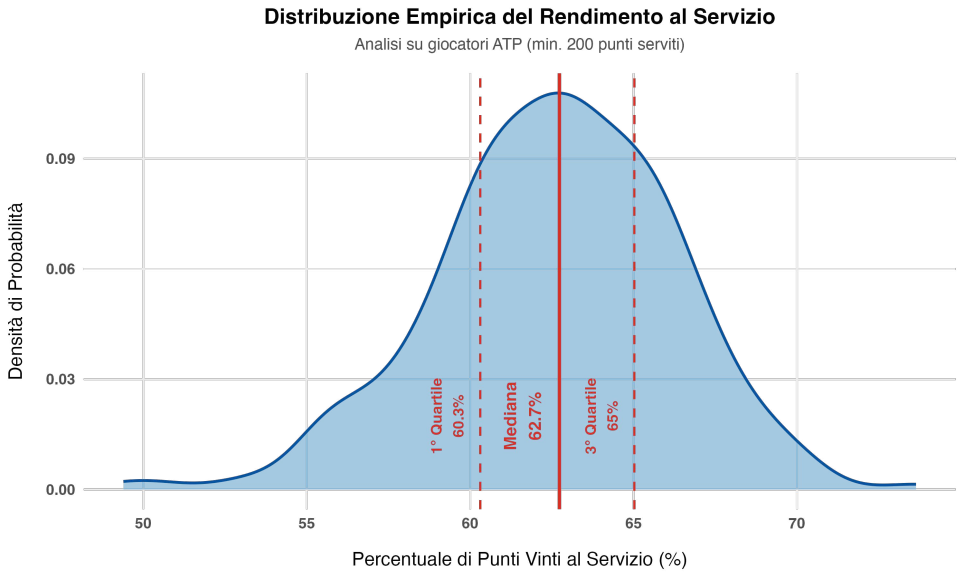


Figura 3.4: Distribuzione empirica del rendimento al servizio per i giocatori del circuito ATP.

Se si limita l'analisi agli scambi giocati di almeno 3 colpi (escludendo quindi gli *ace*, i doppi falli e i servizi senza risposta), il numero mediano di punti vinti al servizio scende

[1] Circuito ATP.

leggermente al 62,6% (Figura 3.5).

Entrambi i grafici mostrano l'evidente vantaggio del giocatore al servizio, vantaggio tratto proprio dal servizio stesso che consente di effettuare fin da subito un primo attacco e costringere l'avversario a doversi difendere. La percentuale di giocatori con rendimento al servizio sotto il 50%, infatti, si attesta a 0,36% nel dataset completo e 0,66% nel dataset filtrato dei soli scambi con almeno 3 colpi, confermando che la quasi totalità dei giocatori del *tour* traggono beneficio dal servizio.

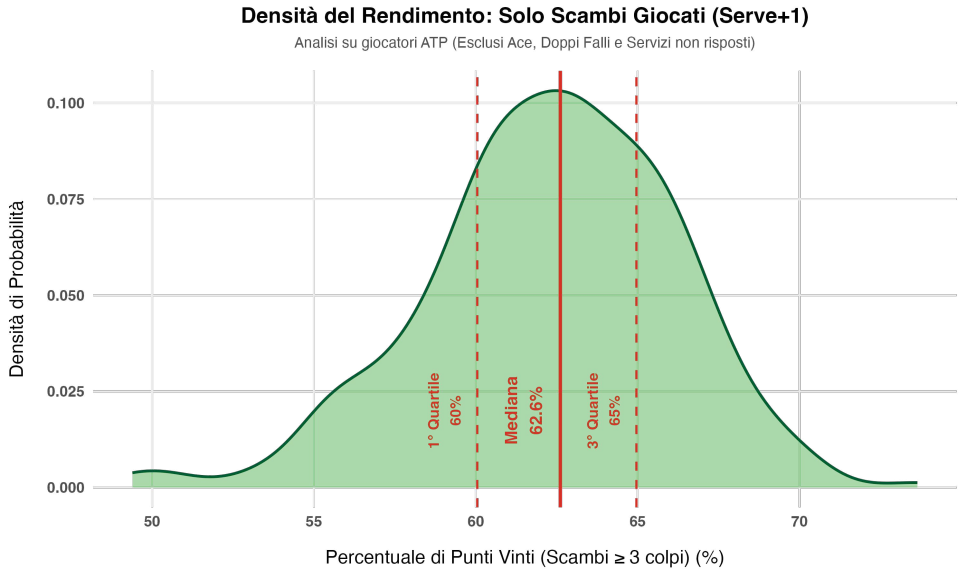


Figura 3.5: Densità del rendimento al servizio nel circuito ATP: analisi limitata ai soli scambi di almeno 3 colpi.

Un grafico finale illustra le scelte spaziali relative al terzo colpo (ovvero il primo colpo del battitore dopo la risposta dell'avversario, indicato come modello *serve + 1*). Le frequenze percentuali delle direzioni (sinistra, corpo, destra) dei terzi colpi di servizio dipendono in larga misura dalla direzione del servizio precedente.

L'osservazione visiva dimostra come la costruzione del punto differisca notevolmente: dopo un primo servizio riuscito, Tsitsipas riceve solitamente una risposta più corta o più difensiva, che gli consente di posizionarsi in modo da prendere l'iniziativa nel correre intorno alla palla e dettare lo scambio, mirando così il terzo colpo principalmente ai lati del campo (per chiudere il punto o consolidare il suo vantaggio territoriale) (47,0% a destra, 39,2% a sinistra). Dopo un secondo servizio, la distribuzione delle direzioni del terzo colpo mostra differenze notevoli. Poiché Tsitsipas deve affrontare ritorni che sono, in media, più lunghi e più aggressivi, ha meno tempo per prepararsi a uno schema offensivo specifico. Sebbene Tsitsipas mantenga percentuali elevate per i terzi colpi a destra (44,0%) e a sinistra (33,5%), l'aumento del numero di ritorni più aggressivi lo costringe a utilizzare un colpo di contenimento/neutro, aumentando così la percentuale di terzi colpi al corpo raggiungendo

il 22,5%.

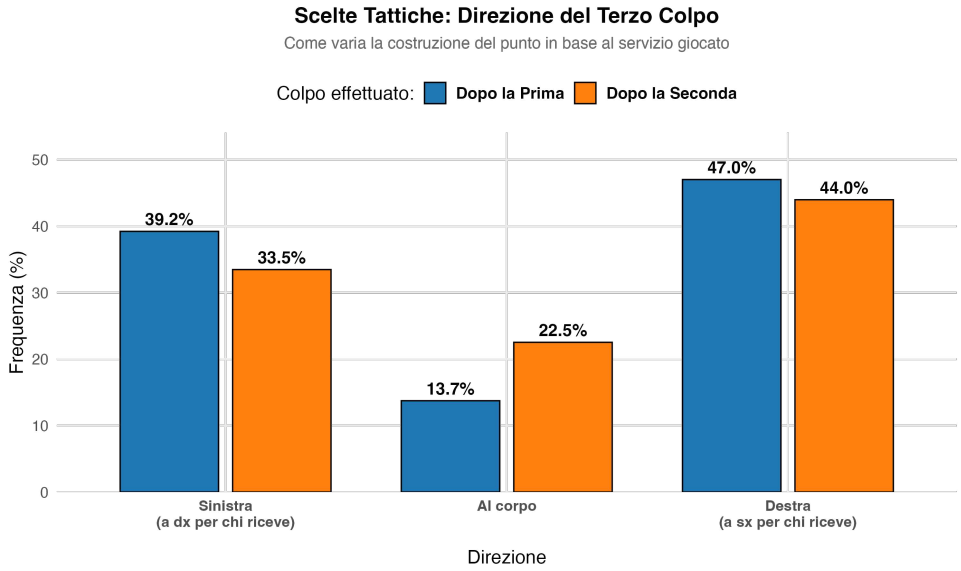


Figura 3.6: Scelte spaziali per la direzione del terzo colpo (*serve + 1*) condizionate all'utilizzo della prima o della seconda palla di servizio.

Pertanto, queste statistiche descrittive forniscono un'utile rappresentazione visiva di quello che Tsitsipas fa normalmente in campo; tuttavia, in senso statistico, la semplice frequenza di utilizzo di una strategia tattica non è necessariamente correlata alla sua efficacia (cioè alla probabilità marginale di vincere il punto). Per approfondire la questione e determinare l'impatto di ogni singola scelta geometrica in uno scambio, indipendentemente dalla moltitudine di fattori di confondimento coinvolti, è necessario ricorrere a modelli multivariati. Questi saranno affrontati e sviluppati nei prossimi capitoli.

Capitolo 4

Metodologia e modelli predittivi

Dopo aver illustrato le variabili considerate, si descrive il processo di costruzione dei modelli predittivi per valutare l'importanza della tattica nelle sequenze a tre colpi. L'approccio adottato parte da una visione generale per poi approfondire i dettagli, mantenendo sempre un legame con la realtà del gioco pratico. Questo capitolo illustra le scelte metodologiche adottate per delimitare il campo di studio, convalidare i risultati nel tempo e sviluppare i modelli, dai più semplici fino agli algoritmi più complessi. Particolare attenzione è stata posta nell'evitare sovradattamento e nel calibrare ogni modello in modo specifico per il giocatore scelto. Nello specifico, devono essere regolati degli iperparametri per gestire il compromesso varianza-distorsione (ad es., Azzalini e Scarpa 2012).

4.1 Definizione del dominio

Dal momento che il 2020 è stato un anno anomalo caratterizzato della pandemia di SARS-CoV-2, con il conseguente calo drastico del numero di tornei disputati durante la stagione, si è deciso di escludere il 2020 dalle analisi e viene ridefinita la finestra temporale dei dati oggetto di studio da gennaio 2021 a giugno 2025.

4.1.1 Cambio di prospettiva: focal player vs opponent

Prima di definire gli effetti non lineari, è stata effettuata una fondamentale trasformazione del dataset. I dati grezzi, originariamente strutturato in ottica "Battitore vs Risponditore", sono stati traslati nella prospettiva del giocatore focale (es. Tsitsipas in questo caso). Mantenere covariate come `fatigue_server` avrebbe generato un irrisolvibile problema di *confounding*, poiché il significato della variabile (stanchezza di Sinner o dell'avversario) si sarebbe invertito ad ogni game. Sono state perciò ingegnerizzate le varianti focali (es. `fatigue_focal_cum` e `fatigue_opp_cum`), garantendo che venga modellato costantemente l'effetto della condizione del giocatore target sulle proprie probabilità di vittoria.

4.1.2 Focus sui turni al servizio

Inizialmente, si è ipotizzato che la costruzione di un unico modello per ciascun giocatore, che considerasse congiuntamente sia i punti al servizio sia quelli in risposta, fosse sufficiente. Per stimare le probabilità di vittoria del punto in entrambe le situazioni e portare l'informazione relativa al vantaggio che ogni giocatore ha nei turni al servizio, è stato impiegato un modello di regressione basato sui dati delle stagioni precedenti, come spiegato nel capitolo 4, al fine di garantire l'aderenza alle dinamiche reali del gioco.

Tuttavia, è stato osservato che la combinazione dei punti al servizio e di quelli in risposta generava ambiguità, poiché un colpo incrociato eseguito da chi è al servizio assume un significato differente rispetto a quello effettuato da chi è in risposta, rendendo molto difficile l'interpretazione finale. Di conseguenza, l'analisi si è concentrata esclusivamente sui punti in cui il giocatore è al servizio, al fine di comprendere con maggiore precisione l'influenza delle scelte di direzione sull'esito del punto e ridurre il rumore di fondo. In questo contesto, la probabilità storica viene utilizzata come riferimento, mentre le variabili relative ai colpi descrivono l'andamento del punto.

Questa scelta non costituisce una limitazione tecnica, ma rappresenta una decisione metodologica consapevole. È stato preferito approfondire l'analisi, anche a costo di restringere il campo di studio, piuttosto che adottare un approccio meno approfondito. Successivamente, verrà illustrato come questo metodo possa essere adattato anche per individuare i punti deboli di un avversario o per identificare aree di miglioramento nei turni di risposta.

4.2 Il sistema di convalida a finestra mobile

Il tennis presenta una marcata componente stagionale, determinata dalle diverse superfici e condizioni di gioco (ad esempio, *indoor* e *outdoor*) che si alternano in specifici periodi dell'anno. Inoltre, la dimensione temporale è rilevante, poiché per effettuare previsioni future sono disponibili solo le informazioni passate. Una selezione casuale dei dati per la fase di test potrebbe introdurre una distorsione, consentendo al modello di prevedere eventi passati utilizzando informazioni provenienti dal futuro.

Per evitare tale distorsione, è stata implementata una convalida a finestra mobile (*walk-forward*). I dati sono stati suddivisi in blocchi temporali sequenziali, adattando il modello ai dati relativi a un periodo di 12 mesi e verificandolo esclusivamente sulle partite dei 6 mesi successivi, spostando progressivamente in avanti le finestre temporali, di sei mesi in sei mesi. Tale approccio consente al modello di essere adattato su una stagione completa e di catturare la variabilità associata alla stagionalità.

Questo metodo presuppone che, nell'arco di 18 mesi, le abilità e le condizioni fisiche degli atleti rimangano costanti. Sebbene tale assunzione non rifletta sempre la realtà, rappresenta una buona approssimazione e un compromesso tra l'utilizzo di dati troppo

datati, che potrebbero non essere rappresentativi delle condizioni attuali degli atleti, e l'impiego di un campione troppo ristretto, che aumenterebbe il rischio di sovradattamento.

L'obiettivo non è individuare un modello universale valido per un lungo periodo, ma calcolare la media degli errori su tutte le finestre temporali per valutare la capacità di adattamento e identificare la famiglia di algoritmi più adeguata all'apprendimento dei comportamenti recenti. L'algoritmo che ottiene le migliori prestazioni medie rappresenta la scelta più affidabile per le previsioni future.

In un contesto applicativo, una volta selezionata la famiglia di modelli con migliori prestazioni, il modello verrebbe riadattato ai dati degli ultimi 12 mesi per effettuare previsioni sulla partita successiva.

In questo modo è possibile valutare se il modello è effettivamente in grado di reagire ai cambiamenti. Per misurare le prestazioni si utilizza l'AUC, area sotto la curva ROC: valori più elevati indicano una maggiore capacità del modello di prevedere correttamente partite non ancora osservate. Si ricerca, quindi, una regola di classificazione in cui la curva ROC si posizioni il più in alto possibile al di sopra della diagonale (ad es., Azzalini e Scarpa 2012).

4.3 Modelli di riferimento (*baseline*)

Per misurare il contributo di modelli dalla struttura più complessa, sono stati definiti e stimati tre modelli di riferimento a complessità crescente, inseriti all'interno del medesimo processo di convalida *walk-forward*.

4.3.1 Base pre-match: Modello "Solo Dixon-Coles"

Il primo modello di riferimento mira a quantificare il limite superiore della prevedibilità di un punto conoscendo esclusivamente la gerarchia di forza dei due giocatori prima dell'inizio dell'incontro.

Dal punto di vista della teoria economica delle scommesse sportive, questo modello rappresenta la "conoscenza a priori" (Dixon e Coles 1997). La variabile `Player_Baseline_Prob`, stimata tramite penalizzazione Ridge sui dati dell'anno solare precedente (si veda il capitolo precedente), condensa il vantaggio strutturale intrinseco del tennista. Il modello è formulato come una regressione logistica univariata:

$$\text{logit}(P(Y_k = 1)) = \beta_0 + \beta_1(\text{Player\_Baseline\_Prob}_k) \tag{4.1}$$

Il confronto tra l'AUC di questo modello univariato e i successivi modelli multivariati permetterà di rispondere a un quesito fondamentale: *quanto le contingenze in-match (stanchezza accumulata, importanza del punto, superficie, ecc.) spostano gli equilibri rispetto alla pura probabilità pre-match?*

4.3.2 Regressione logistica standard

Per valutare il valore aggiunto dell'estrazione di regole non lineari e interazioni, è opportuno confrontare tali metodi con un modello parametrico standard come lo sono i modelli lineari generalizzati (GLM), formulati secondo il *framework* classico di Nelder e Wedderburn (1972).

In questo modello, tutti i predittori validi disponibili al momento del servizio vengono inseriti come effetti principali. La probabilità di vincere il punto è modellata assumendo una rigorosa indipendenza tra le osservazioni e una relazione strettamente lineare sulla scala del logit:

$$\text{logit}(P(Y_k = 1)) = \beta_0 + \sum_{m=1}^{M} \beta_m X_{mk} \tag{4.2}$$

Il modello logistico classico, non potendo flettere la propria forma funzionale né dividere lo spazio delle variabili in sottogruppi condizionati, traccia come confine di decisione un iperpiano. Qualora le prestazioni del GLM risultassero equivalenti a quelle di modelli più complessi, significherebbe che le dinamiche tennistiche incluse nel dataset impattano sull'esito del punto in modo additivo e proporzionale, rendendo superflua l'adozione di modelli non parametrici.

4.3.3 Modello logistico misto

Sapendo che il tennis è uno sport le cui osservazioni (i punti) sono raggruppate all'interno di set e partite, assumere la totale indipendenza degli eventi (come fa il GLM) viola un'assunzione statistica di base (Breslow e Clayton 1993).

Per questo motivo, la regressione logistica è stata estesa in un Generalized Linear Mixed Model (GLMM), implementato tramite il pacchetto `lme4` (Bates et al. 2015). La formula include la stessa componente fissa del GLM, alla quale si aggiungono dei termini di intercetta casuale per modellare la varianza gerarchica:

$$\text{logit}(P(Y_{ijk} = 1)) = \beta_0 + \sum_{m=1}^{M} \beta_m X_{mijk} + u_{0j} + v_{0jk} \tag{4.3}$$

dove Y_{ijk} rappresenta l'esito (0 perso, 1 vinto) dell'i-esimo punto del k-esimo *set* all'interno della j-esima partita, $u_{0j} \sim \mathcal{N}(0, \sigma^2_{\text{match}})$ rappresenta l'effetto casuale della j-esima partita e $v_{0jk} \sim \mathcal{N}(0, \sigma^2_{\text{set}})$ rappresenta l'effetto casuale del k-esimo *set* annidato nella j-esima partita.

Giustificazione delle Scelte Modellistiche per il GLMM

La stima di un GLMM su dataset sportivi ad alta granularità pone significative sfide computazionali e matematiche. Per garantire stime convergenti e rigorose, sono state prese

due decisioni:

- Esclusione del livello *game* (`unique_game_id`): Inizialmente, la gerarchia teorica del tennis prevederebbe la nidificazione dei punti anche all'interno del singolo game. Tuttavia, essendo un game composto in media da 6-8 punti, il tentativo di stimare una varianza specifica per gruppi così esigui, in presenza di un centinaio di effetti fissi (considerando le variabili indicatrici delle qualitative), porta matematicamente l'algoritmo verso un fenomeno di adattamento singolare, cioè ad una varianza stimata che collassa a zero. L'esclusione del livello game assicura che gli effetti casuali siano calcolati su gruppi sufficientemente ampi (circa 60 punti per *set* e 150 per *match*), garantendo stabilità alle matrici di covarianza.

- Approssimazione di Laplace (`nAGQ = 1`): Avendo rimosso l'instabilità dei micro-cluster (i *game*), è stato possibile impiegare l'integrazione numerica tramite Approssimazione di Laplace per il calcolo della verosimiglianza marginale (Breslow e Clayton 1993), rispetto a metodi di approssimazione più veloci ma meno accurati (come la penalizzazione zero-point, `nAGQ=0`).

Il confronto finale in fase di convalida *walk-forward* tra il GLMM (che è un modello gerarchico parametrico lineare) e il GPBoost (che è un modello gerarchico basato su alberi non lineari e che verrà approfondito in seguito nel paragrafo 4.6.2) permetterà di affermare con certezza matematica se i vantaggi del Boosting derivino unicamente dalla gestione della varianza di gruppo o dalla reale capacità degli alberi di scovare interazioni tattiche profonde impossibili o difficili da modellare parametricamente.

4.4 Generalized Additive Mixed Models (GAMM)

Al fine di valutare le prestazioni dei modelli capaci di modellare esplicitamente le relazioni non lineari, è stato implementato un generalized additive mixed model (GAMM). Questa famiglia di modelli estende i modelli lineari generalizzati (GLM) rilassando l'assunzione di linearità per i predittori continui tramite l'uso di funzioni di lisciamento (spline) (Hastie e Tibshirani 1990) e, parallelamente, accomoda la struttura correlata dei dati tramite l'inclusione di effetti casuali (random effects) (Wood 2017).

4.4.1 Fondamenti teorici e scelta dell'algoritmo

La stima di un modello additivo misto per dati non gaussiani (come nel caso della variabile risposta binaria per l'esito del punto) presenta notevoli sfide computazionali e teoriche. La formulazione originaria dei GAMM, implementata nella funzione nativa `gamm()` del pacchetto R `mgcv`, si basa sulla *penalized quasi-likelihood* (PQL) introdotta da Lin e Zhang (1999). Tuttavia, la letteratura statistica ha ampiamente dimostrato come l'approccio PQL,

basandosi su un'approssimazione in serie di Taylor al primo ordine, tenda a prodrre una sottostima (attenuazione) dei parametri di varianza degli effetti casuali quando applicato a dati strettamente binari (prove di Bernoulli, dove $n = 1$ per ogni riga, come nel caso del singolo punto giocato) (Breslow e Clayton 1993).

Oltre alla distorsione matematica, l'elevato volume di dati trattati nel presente studio (oltre 400.000 osservazioni) rende l'uso della PQL tramite la funzione standard `gamm()` computazionalmente proibitivo in termini di memoria allocata e tempi di esecuzione.

Per questi due motivi, in accordo con le dimostrazioni di Wood (2011), si è scelto di abbandonare l'approccio PQL in favore dell'ottimizzazione diretta della *Restricted Marginal Likelihood* (REML) tramite approssimazione di Laplace. Al fine di garantire la convergenza e tempi di calcolo compatibili con l'architettura di validazione *walk-forward*, il modello è stato stimato utilizzando la funzione `bam()` (*Big Additive Models*). Questa implementazione sfrutta il metodo *Fast REML* (`fREML`), che non solo gestisce l'ottimizzazione matriciale su *large datasets* in modo efficiente, ma ottimizza analiticamente i parametri di penalizzazione delle *spline* (λ), riducendo drasticamente il rischio di *overfitting* tipico dei tradizionali metodi basati su *Generalized Cross-Validation* (GCV) in presenza di dati correlati (Wood, Goude, e Shaw 2015).

4.4.2 Specificazione del modello per i dati tennistici

La costruzione della formula del modello ha richiesto scelte rigorose per riflettere le dinamiche fisiche e tattiche del tennis, declinate secondo i framework teorici proposti da Pedersen et al. (2019) per i modelli gerarchici additivi (HGAM).

Struttura gerarchica (effetti casuali) nel framework additivo

Seguendo le raccomandazioni di Pedersen et al. (2019), la dipendenza gerarchica e temporale dei dati è stata modellata includendo la partita e il *set* come effetti casuali. Essendo all'interno della specificazione di un Modello Additivo Generalizzato Misto (GAMM), il predittore lineare η_i per l'osservazione i-esima viene esteso per includere sia componenti non parametriche sia effetti casuali, assumendo la seguente forma:

$$g(\mu_i) = \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \sum_{k=1}^{K} f_k(z_{ki}) + b_{m[i]} + b_{s[i]} \tag{4.4}$$

dove:

- $g(\cdot)$ rappresenta la funzione legame (*link function*) logistica;
- $\mathbf{x}_i^\top \boldsymbol{\beta}$ è la componente strettamente parametrica degli effetti fissi;
- $f_k(z_{ki})$ sono le funzioni di lisciamento (*smooth functions*) centrate applicate a K covariate continue z_k;

- $b_{m[i]}$ e $b_{s[i]}$ sono gli effetti casuali associati rispettivamente alla partita m-esima e al *set* s-esimo a cui appartiene l'osservazione i.

Seguendo l'impostazione metodologica descritta da Wood (2017) e la nota equivalenza tra modelli misti e lisciamento penalizzato (Ruppert, Wand, e Carroll 2003), gli effetti casuali b_m e b_s sono stati formalizzati e stimati all'interno dello stesso paradigma delle *splines* utilizzato per le funzioni f_k. Nello specifico, l'inclusione di un'intercetta casuale per una variabile categorica è matematicamente equivalente all'introduzione di una base funzionale associata a una penalizzazione di tipo *Ridge* (ℓ_2).

Assumendo che $b_m \sim N(0, \sigma_m^2)$ e $b_s \sim N(0, \sigma_s^2)$, il modello GAMM viene adattato massimizzando una log-verosimiglianza penalizzata ℓ_p, dove il termine complessivo di penalizzazione $\mathcal{P}$ regola sia la ruvidità delle curve continue sia la varianza degli effetti casuali:

$$\mathcal{P} = \sum_{k=1}^{K} \lambda_k \int [f_k''(z)]^2 dz + \lambda_m \sum_m b_m^2 + \lambda_s \sum_s b_s^2 \tag{4.5}$$

I parametri di lisciamento (*smoothing parameters*) λ_m e λ_s controllano il grado di contrazione (*shrinkage*) verso lo zero e sono inversamente proporzionali alle componenti di varianza degli effetti casuali (es. $\lambda_m \propto 1/\sigma_m^2$).

Per garantire stime non distorte dei molteplici parametri di lisciamento e delle componenti di varianza, il modello è stato stimato ricorrendo alla Massima Verosimiglianza Ristretta (REML). La stima REML, infatti, corregge la distorsione verso il basso tipica della stima di massima verosimiglianza ordinaria tenendo conto dei gradi di libertà persi nella stima dei parametri fissi, risultando il metodo più robusto per i modelli GAMM.

L'inclusione di livelli gerarchici più granulari, come il singolo gioco (`unique_game_id`), è stata metodologicamente esclusa per gli stessi motivi descritti in precedenza.

Effetti non-lineari e gestione delle interazioni

Per le variabili continue aventi un numero sufficiente di valori unici (es. probabilità di base di Dixon-Coles), sono state impiegate delle *thin plate splines* (`bs="tp"`). A differenza di altre basi, queste non richiedono la pre-specificazione della posizione dei nodi. Al fine di evitare la singolarità della matrice causata dalla scarsità dei dati in alcuni regimi estremi, la dimensione della base (i gradi di libertà massimi) è stata vincolata in modo prudente a $k = 5$. Sarà poi la penalizzazione `fREML` a ridurre i gradi di libertà equivalenti (EDF) estraendo la reale forma funzionale, lineare o non-lineare.

4.4.3 Implementazione predittiva nella *walk-forward*

L'uso di un modello ad effetti misti all'interno di una convalida *walk-forward* introduce una criticità strutturale: il *set* di verifica (composto dal semestre temporalmente successivo

all'addestramento) conterrà inevitabilmente partite mai viste dal modello nella fase di training. Questo impedisce all'algoritmo di conoscere l'effetto casuale per quel nuovo `match_id`. Per aggirare questo ostacolo matematico, si è sfruttato l'argomento `exclude` della funzione `predict.bam()`: imponendo al modello di ignorare i termini `s(match_id)` in fase predittiva, le probabilità dei punti futuri vengono stimate sulla *media di popolazione* del giocatore, depurata da specifici effetti-partita pregressi.

4.5 Modellazione ad albero interpretabile: generalized linear mixed-effects model trees (GLMM Trees)

In ambito sportivo, le analisi si dividono principalmente in due macro-aree: quelle con l'obiettivo di migliorare la performance atletico-tattica e quelle relative alle scommesse. Nel secondo caso, chiaramente, è indispensabile massimizzare la capacità predittiva pura, anche a costo di sacrificare totalmente l'interpretabilità del modello (*black-box*). Nel primo caso, che corrisponde all'obiettivo del presente studio, è invece indispensabile poter interpretare i risultati affinché siano utilizzabili per allenatori e analisti.

Per affrontare questo aspetto, si è deciso di adattare un modello *generalized linear mixed-effects model tree* (GLMM Tree) (Fokkema, Edbrooke-Childs, e Wolpert 2021).

4.5.1 Formulazione Teorica e Algoritmo di Stima

L'algoritmo GLMM Trees si fonda sul framework del *model-based recursive partitioning* (MOB). A differenza dei classici alberi CART, che stimano una semplice media o proporzione in ogni nodo foglia, i GLMM Trees stimano un vero e proprio Modello Lineare Generalizzato Misto all'interno della struttura ad albero (Fokkema, Edbrooke-Childs, e Wolpert 2021).

Sia i l'indice della singola osservazione (il punto giocato). La formulazione matematica del modello applicato in questa sede assume la seguente forma:

$$g(\mu_i) = \mathbf{x}_i^\top \boldsymbol{\beta}_{j[i]} + b_{m[i]} \tag{4.6}$$

dove:

- $g(\cdot)$ è la funzione legame (*link function*) logistica per la classificazione binaria della probabilità di vittoria del punto;
- $\mathbf{x}_i^\top \boldsymbol{\beta}_{j[i]}$ rappresenta la componente degli effetti fissi locali. Nello specifico della nostra applicazione, $\boldsymbol{\beta}_{j[i]}$ è il parametro (o vettore di parametri) stimato specificamente per il nodo terminale j (foglia) in cui ricade l'osservazione i. Questo termine rappresenta la probabilità di base di vincere il punto in quel determinato sottogruppo di condizioni tattiche;

- $b_{m[i]}$ rappresenta l'effetto casuale globale, distribuito come $N(0, \sigma_m^2)$, associato al *match* m-esimo a cui appartiene l'osservazione i. Tale effetto rimane costante in tutto l'albero e modella la varianza latente specifica di ogni singola partita.

L'algoritmo procede in modo iterativo alternando due fasi fino a convergenza, riprendedno la logica dell'algoritmo di *backfitting* (Fokkema, Edbrooke-Childs, e Wolpert 2021):

1. Stima dell'albero condizionata agli effetti casuali: la struttura dell'albero viene stimata tramite MOB, inserendo le previsioni correnti degli effetti casuali ($b_{m[i]}$) come *offset* nel predittore lineare. Dal punto di vista operativo, in questa fase l'algoritmo effettua il partizionamento lavorando sui residui parziali lasciati inspiegati dalla componente gerarchica. Il partizionamento si ferma in base a formali test statistici di fluttuazione dei parametri, prevenendo la distorsione da selezione delle variabili.

2. Stima degli effetti casuali condizionata all'albero: i parametri strutturali e la componente di varianza dell'effetto casuale vengono aggiornati massimizzando la verosimiglianza penalizzata (sfruttando il motore di calcolo del pacchetto `lme4` (Bates et al. 2015)). Durante questo step, la struttura dell'albero e le stime dei parametri locali ($\boldsymbol{\beta}_{j[i]}$) ottenute al punto precedente vengono mantenute fisse e passate a loro volta come *offset*.

Questo processo di stima alternata (che modella iterativamente i residui dell'ultimo passaggio) continua fino a quando la struttura dell'albero si stabilizza e si raggiunge la convergenza della log-verosimiglianza congiunta.

4.5.2 Scelte metodologiche e implementazione pratica

L'applicazione del modello ai dati tennistici ha richiesto decisioni metodologiche per garantire la stabilità numerica e la corretta convalida predittiva.

Definizione degli effetti casuali e singular fit

In fase di specificazione del modello, si è tentato inizialmente di sfruttare appieno la struttura gerarchica dei dati includendo effetti casuali annidati sia per il match (`match_id`) sia per i *set* giocati all'interno di esso (`unique_set_id`). Tuttavia, l'ottimizzatore ha restituito un *boundary (singular) fit warning*. Da un punto di vista statistico, ciò indica che la stima della varianza intra-match spiegata dai singoli *set*, condizionata alla struttura dell'albero, è collassata a zero. È probabile che i predittori includessero buona parte dell'informazione di `unique_set_id`. Mantenere componenti di varianza nulle in modelli complessi porta a sovra-parametrizzazione e instabilità numerica. Di conseguenza, si è optato per una formula più parsimoniosa, mantenendo unicamente l'intercetta casuale a livello di partita.

Prevenzione del data leakage (fattori non visti)

Gli alberi condizionati non ammettono l'estrapolazione di probabilità per categorie di variabili nominali mai osservate nel *training set*. In un'architettura di convalida *walk-forward*, il manifestarsi di nuove modalità (es. un nuovo torneo o una nuova tipologia di superficie nel *test set*) causerebbe il blocco delle previsioni. Per evitare ogni forma di *data leakage*, è stata implementata una funzione che identifica le modalità sconosciute nei dati di test declassandole dinamicamente a una macro-categoria neutra mancante, calcolata unicamente in fase di stima. Questo garantisce stime eque dell'errore di generalizzazione in uno scenario produttivo reale.

Regolazione raggruppata e inferenza a livello di popolazione

Il controllo della crescita dell'albero è stato effettuato esplorando due parametri fondamentali: il livello di significatività per i test di stabilità (`alpha`) e la profondità massima dell'albero (`maxdepth`). La regolazione di queste due quantità è stata valutata su una suddivisione dei dati in stima e verifica interna basata sul raggruppamento per `match_id`, per preservare l'indipendenza delle osservazioni. Infine, poiché il compito finale è la previsione su partite future (i cui effetti casuali specifici sono ignoti a priori), in fase di verifica l'algoritmo è stato forzato a calcolare le previsioni ignorando gli effetti casuali (`re.form = NA`), producendo così stime puramente basate sulle regole estratte dall'albero.

4.6 Metodi di boosting: indipendenza vs. effetti misti

Per massimizzare la precisione del modello e descrivere relazioni complesse che non si conformano a relazioni lineari, l'attenzione si è spostata dai modelli costituiti da un solo albero all'uso di modelli di tipo *ensemble*. Nel tempo, le tecniche di *ensemble* hanno preso due grandi orientamenti: il *Bagging*, come la Random Forest, che volge particolare cura a ridurre la varianza facendo crescere alberi profondi e indipendenti fra loro, e il *Boosting* (Freund e Schapire 1997), il quale mira a ridurre la distorsione adattando ai dati sequenzialmente alberi piccoli, in modo che ad ogni iterazione vengano corretti gli errori lasciati dalla fase precedente. Per ragioni tecniche in questo studio ci si è concetrati sul *boosting*, tuttavia sarebbe interessante un confronto con il *bagging* adattando, ad esempio, un modello MERF.

Da un lato, il pacchetto `gpboost` (Sigrist 2022) allo stato attuale supporta la stima dei parametri di covarianza unicamente tramite aggiornamenti di tipo *gradient boosting* (GBDT) e non per diramazione parallela e autonoma. Dall'altro, gli algoritmi nativi basati sul metodo *Expectation-Maximization* (EM) usati da Hajjem, Bellavance, e Larocque (2014) non trovano ad oggi un'implementazione stabile e accettabile per estendere il principio ai modelli generalizzati (GMERF) per la classificazione binaria con funzione di *link logit*. Di

conseguenza, per garantire il rigore metodologico e la stabilità numerica nella stima degli effetti casuali, l'architettura si è concentrata sull'unica corrente percorribile: il *Gradient Boosting*.

4.6.1 Extreme Gradient Boosting (XGBoost)

Il primo modello di *ensemble* implementato è l'Extreme Gradient Boosting (XGBoost), introdotto nella letteratura scientifica da Chen e Guestrin (2016) come evoluzione altamente scalabile della *Gradient Boosting Machine* di Friedman (2001).

L'algoritmo minimizza il seguente funzionale obiettivo $\mathcal{L}^{(t)}$, costituito da una funzione di perdita differenziabile ed un termine di regolarizzazione Ω che penalizza la complessità dell'albero:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t) \tag{4.7}$$

dove:

- $l(\cdot, \cdot)$ è la funzione di perdita che misura lo scostamento tra il valore reale osservato y_i e la previsione del modello. Nello specifico della classificazione binaria, l corrisponde alla *log-loss* (entropia incrociata binaria);
- $\hat{y}_i^{(t-1)}$ rappresenta la previsione per l'osservazione i-esima accumulata dal modello fino all'iterazione precedente $(t-1)$;
- $f_t(\mathbf{x}_i)$ è il contributo predittivo del nuovo albero decisionale aggiunto all'iterazione corrente t, valutato per il vettore di covariate $\mathbf{x}_i$;
- $\Omega(f_t)$ è il termine di regolarizzazione strutturale. Per rendere esplicita la penalizzazione, esso è definito matematicamente come:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{4.8}$$

  in cui T indica il numero totale di foglie (nodi terminali) dell'albero f_t, w_j rappresenta il peso (o punteggio predittivo) associato alla foglia j-esima, γ penalizza la creazione di nuovi nodi e λ controlla la penalizzazione basata sulla norma L_2 dei pesi ($\sum w_j^2$), limitandone la magnitudo per prevenire il sovradattamento.

XGBoost gode, inoltre, di un vantaggio cruciale per i dati in esame rispetto ad altri algoritmi: esso, infatti, è capace di gestire i valori mancanti (denotati con `NA`) in modo nativo, senza ricorrere ad alcuna imputazione. È l'algoritmo medesimo ad apprendere la direzione ottimale quando, durante il partizionamento, si trova al cospetto di un'osservazione con valori mancanti, massimizzando il guadagno nella funzione obiettivo.

Ottimizzazione e prevenzione del data leakage. I parametri di regolazione sono la massima profondità degli alberi (`max_depth` $\in \{1, 2, 3, 4\}$) e il tasso di apprendimento o *shrinkage* (`eta` $\in \{0.01, 0.05, 0.1, 0.2, 0.3\}$). Per evitare il sovradattamento, il numero ottimale di alberi si determina mediante una procedura di arresto anticipato (*early stopping*) su un insieme di convalida interno (70% insieme di stima, 30% convalida). La stratificazione delle singole suddivisioni di convalida è stata effettuata per `match_id`. Infine, individuati i valori ottimali dei parametri e del numero ideale di iterazioni, il modello finale è stato riadattato all'intero insieme di stima.

4.6.2 GPBoost: Gradient Boosting con Effetti Misti

Il principale limite teorico di XGBoost proviene dall'assunzione di indipendenza dei campioni (i.i.d.). Nel tennis, i punti relativi allo stesso match, set o game condividono una varianza stocastica difficilmente prevedibile dalle sole variabili esogene.

Questo limite è stato superato con l'adozione di GPBoost (Sigrist 2022), un metodo all'avanguardia che unisce l'efficacia predittiva del *tree-boosting* ai vantaggi della modellazione ad effetti misti. In contrasto con le metodiche fondate sull'algoritmo *Expectation-Maximization*, che ricavano la stima della foresta e degli effetti casuali mediante approssimazione alternata con il rischio di partizioni sub-ottimali, GPBoost fa leva su un elegante metodo di ottimizzazione congiunta. L'algoritmo integra analiticamente gli effetti casuali definendo la funzione di verosimiglianza marginale. Di conseguenza, il calcolo del gradiente e dell'hessiano, necessario alla costruzione degli alberi, incorpora automaticamente la matrice di covarianza: la struttura di correlazione viene appresa dall'albero nel momento stesso in cui viene effettuato una partizione.

Base gerarchica. Mentre il modello ad albero singolo (GLMM Trees, basato sul pacchetto `lme4`) generava instabilità (*singular fit*) nel valutare la varianza annidata tra i *set*, costringendo a limitare l'effetto casuale unicamente a livello di *match*, GPBoost riesce a gestire l'intera orditura gerarchica del tennis senza incertezze.

Ottimizzazione specifica e stabilità numerica. La superiorità matematica di GPBoost si è manifestata chiaramente in fase di implementazione pratica, gestendo senza difficoltà l'intera gerarchia annidata (`match_id` $\rightarrow$ `unique_set_id` $\rightarrow$ `unique_game_id`). Da un punto di vista formale, il modello GPBoost per la classificazione binaria estende il *framework* dei modelli misti sostituendo la componente parametrica degli effetti fissi con un insieme di alberi decisionali:

$$g(\mu_i) = F(\mathbf{x}_i) + \mathbf{z}_i^\top \mathbf{b} \tag{4.9}$$

dove $g(\cdot)$ è la funzione legame logistica, $\mathbf{z}_i^\top \mathbf{b}$ modella gli effetti casuali della struttura gerarchica e $F(\mathbf{x}_i) = \sum_{t=1}^{K} f_t(\mathbf{x}_i)$ rappresenta la somma dei contributi non lineari degli alberi di *boosting*.

Durante l'addestramento, la complessità di ciascun nuovo albero f_t viene controllata da un termine di regolarizzazione strutturale $\Omega(f_t)$, applicato ai punteggi (o pesi) w_j delle sue foglie:

$$\Omega(f_t) = \gamma T + \lambda_1 \sum_{j=1}^{T} |w_j| + \frac{1}{2}\lambda_2 \sum_{j=1}^{T} w_j^2 \tag{4.10}$$

in cui T è il numero totale di nodi terminali, mentre λ_1 e λ_2 rappresentano i parametri che governano rispettivamente l'intensità della penalizzazione basata sulla norma L_1 (Lasso, ovvero la somma dei valori assoluti dei pesi) e sulla norma L_2 (Ridge, ovvero la somma dei quadrati).

Per massimizzare le performance di questa architettura, è stata implementata una procedura di scelta dei parametri di regolazione dedicata e fortemente regolarizzata. Tramite una griglia di ricerca valutata su uno insieme di convalida (sempre stratificato per *match*), sono stati esplorati la profondità massima (`max_depth` $\in \{1, 2, 3\}$), il tasso di apprendimento (`learning_rate` $\in \{0.01, 0.05\}$), la percentuale di variabili campionate per ogni albero (`feature_fraction` $\in \{0.6, 0.8, 1.0\}$) e il numero minimo di osservazioni per foglia (`min_data_in_leaf` $\in \{20, 50, 100\}$).

Per prevenire il sovradattamento intrinseco nei modelli gerarchici, la stima della componente ad albero è stata vincolata fissando preventivamente i suddetti parametri di penalizzazione ai pesi ($\lambda_1 = 1$ e $\lambda_2 = 5$), affiancati da un meccanismo di arresto anticipato basato sulla massimizzazione dell'AUC con una tolleranza di 40 iterazioni.

Infine, durante le fasi di predizione sull'insieme di convalida e di verifica (composto da un semestre temporale successivo), sono state bloccate le previsioni degli effetti latenti (`pred_latent = FALSE`, `predict_var = FALSE`). Questa scelta metodologica è risultata cruciale: forzando il modello a calcolare le probabilità unicamente a livello di popolazione (*population-level mean*), è stato possibile valutare oggettivamente l'adattabilità e la robustezza delle regole tattiche estratte dall'algoritmo su partite mai osservate in precedenza.

4.7 Modellazione deep learning: feed-forward neural network (`Keras`)

Al fine di catturare relazioni non lineari complesse all'interno dei dati, è stata implementata una rete neurale artificiale a propagazione in avanti (*feed-forward*) utilizzando la struttura *Keras* (Chollet et al. 2015). Per garantire la robustezza del modello e prevenire il sovradattamento, l'architettura e i parametri di apprendimento sono stati ottimizzati attraverso una rigorosa procedura di regolazione.

Analisi preliminare e standardizzazione

I dati in ingresso sono stati sottoposti ad un'analisi preliminare specifica per le reti neurali. Le variabili identificative di natura gerarchica e ad alta cardinalità, quali `match_id`, `unique_set_id` e `unique_game_id`, sono state escluse dall'insieme di predittori per evitare la mera memorizzazione del contesto e il conseguente rischio di sovradattamento.

I valori mancanti sono stati inputati utilizzando la mediana per le variabili numeriche e una categoria dedicata (*missing*) per i fattori categoriali, assicurando il perfetto allineamento dei livelli tra insieme di stima e di verifica. Dopo aver rimosso i predittori a varianza nulla, le variabili categoriali sono state trasformate in matrici di variabili indicatrici. Infine, i dati sono stati standardizzati; per prevenire qualsiasi forma di *data leakage*, medie e deviazioni standard sono state calcolate unicamente sui dati di stima e, solo in un secondo momento, applicate all'insieme di verifica.

Regolazione dei parametri e *grid search*

La selezione degli iperparametri è avvenuta configurando una griglia di ricerca (*grid search*) su una suddivisione di convalida interna (80% stima, 20% verifica). Al fine di garantire l'indipendenza delle osservazioni e simulare uno scenario predittivo reale, la suddivisione è stata raggruppata in base all'identificativo della partita (`match_id`). Questo espediente metodologico assicura ancora una volta che le metriche di convalida vengano calcolate su interi incontri mai esposti alla rete durante la fase di stima.

Lo spazio di esplorazione per valutare sistematicamente le combinazioni di parametri ha coinvolto le seguenti dimensioni:

- **Capacità della rete e non-linearità:** architettura "large" (tre strati nascosti o *hidden layers* rispettivamente con 128, 64 e 32 neuroni) oppure "small" (64, 32 e 16 neuroni). Tutte le funzioni di attivazione interne sono state impostate al tipo ReLU (*Rectified Linear Unit*). Questa scelta metodologica è lo standard nel *Deep Learning*, poiché la funzione $f(x) = \max(0, x)$ mitiga il problema della scomparsa del gradiente (*vanishing gradient*) e accelera la convergenza dell'algoritmo rispetto alle classiche funzioni tangenti iperboliche o sigmoidee, introducendo al contempo un'utile sparsità nelle attivazioni.

- **Regolarizzazione mediante *dropout*:** al fine di mitigare il rischio di sovradattamento intrinseco nelle reti neurali profonde, è stata applicata la tecnica di regolarizzazione nota come *dropout* (Srivastava et al. 2014). Tale procedura consiste nella disattivazione casuale (con una certa probabilità p) di una specifica frazione di neuroni durante ciascuna iterazione della fase di addestramento. Questo meccanismo impedisce il coadattamento complesso tra i nodi, forzando la rete a distribuire i pesi e ad apprendere rappresentazioni latenti più robuste e generalizzabili. Nello specifico della griglia di

ricerca, sono stati esplorati regimi di *dropout* "high" (con tassi di disattivazione pari a 0.3 per il primo strato nascosto e 0.2 per il secondo) e regimi "low" (tassi pari a 0.2 per il primo strato e 0.1 per il secondo).

- **Tasso di apprendimento (*learning rate*):** esplorato con valori pari a 0.001 e 0.0005 per l'ottimizzatore `Adam` (Kingma e Ba 2014).

Durante la fase di *grid search*, il processo di adattamento è stato regolato tramite il meccanismo di arresto anticipato, monitorando l'entropia incrociata binaria con una tolleranza impostata a 8 "epoche". La combinazione ottimale di iperparametri è stata infine selezionata massimizzando l'area sotto la curva ROC (AUC) misurata sull'insieme di convalida.

Capitolo 5

Risultati e confronti

I risultati dei diversi metodi statistici e di *machine learning* sviluppati nel capitolo 4 sono stati analizzati in questo capitolo e le loro prestazioni predittive sono state valutate utilizzando una procedura di convalida *walk-forward*. I risultati sono stati confrontati utilizzando l'area sotto la curva ROC (AUC) come metrica di riferimento; l'AUC misura la capacità discriminante del modello di classificare correttamente l'esito vittoria-sconfitta di un singolo punto (variabile binaria) in una partita di tennis.

Un valore AUC di 0,50 indica una classificazione completamente casuale. Pertanto, valori più elevati indicano una maggiore capacità del modello di estrarre il segnale dal rumore stocastico presente in uno scambio di tennis. Le medie dei valori di AUC ottenuti durante le diverse finestre temporali di convalida sono mostrate nella tabella 5.1.

Sebbene l'AUC costituisca la metrica principale su cui si fonderà l'analisi — poiché valuta la bontà intrinseca delle probabilità stimate misurando la capacità di *ranking* del modello indipendentemente da uno specifico punto di taglio (*cut-off*) — per ragioni di completezza metodologica e per consentire un confronto più approfondito, le prestazioni sono state valutate anche attraverso due metriche addizionali: l'*F1 Score* e il coefficiente di correlazione di Matthews (MCC). L'*F1 Score* rappresenta la media armonica tra precisione e richiamo (*recall*), mentre l'MCC è una misura di correlazione tra le classificazioni osservate e quelle previste, nota in letteratura per la sua robustezza e affidabilità statistica anche in presenza di classi sbilanciate (Chicco e Jurman 2020). Il coefficiente MCC varia tra -1 e 1; a livello interpretativo, +1 indica una previsione perfetta, 0 indica una previsione casuale (non migliore del caso) e -1 indica una totale discrepanza tra previsione e osservazione.

A differenza dell'AUC, che valuta la capacità discriminante del modello in modo indipendente dalla soglia, il calcolo dell'*F1 Score* e dell'MCC richiede la binarizzazione preventiva delle probabilità stimate. In questo studio, si è ritenuto metodologicamente scorretto adottare una soglia euristica fissa a $0,50$. Considerando lo sbilanciamento strutturale intrinseco del dominio tennistico (in cui il giocatore al servizio vanta una probabilità di base di vittoria fisiologicamente superiore al 60%), l'utilizzo del $0,50$ avrebbe distorto le metriche

di prestazione, alterando la classificazione e sovrastimando i falsi positivi. Pertanto, per ogni iterazione della convalida *walk-forward* e per ciascun modello, le probabilità continue sono state convertite in previsioni binarie determinando dinamicamente la soglia di taglio ottimale. Tale soglia è stata calcolata massimizzando la statistica J di Youden (Youden 1950)

(J = Sensibilità + Specificità − 1) sulla curva ROC delle stime, garantendo così una classificazione rigorosa e aderente alla reale distribuzione campionaria. Le medie di tali metriche sono state integrate all'interno della medesima tabella riepilogativa.

Tabella 5.1: Valori medi di AUC, *F1 Score* e MCC per i diversi algoritmi statistici e di *machine learning* sviluppati nel capitolo 4.

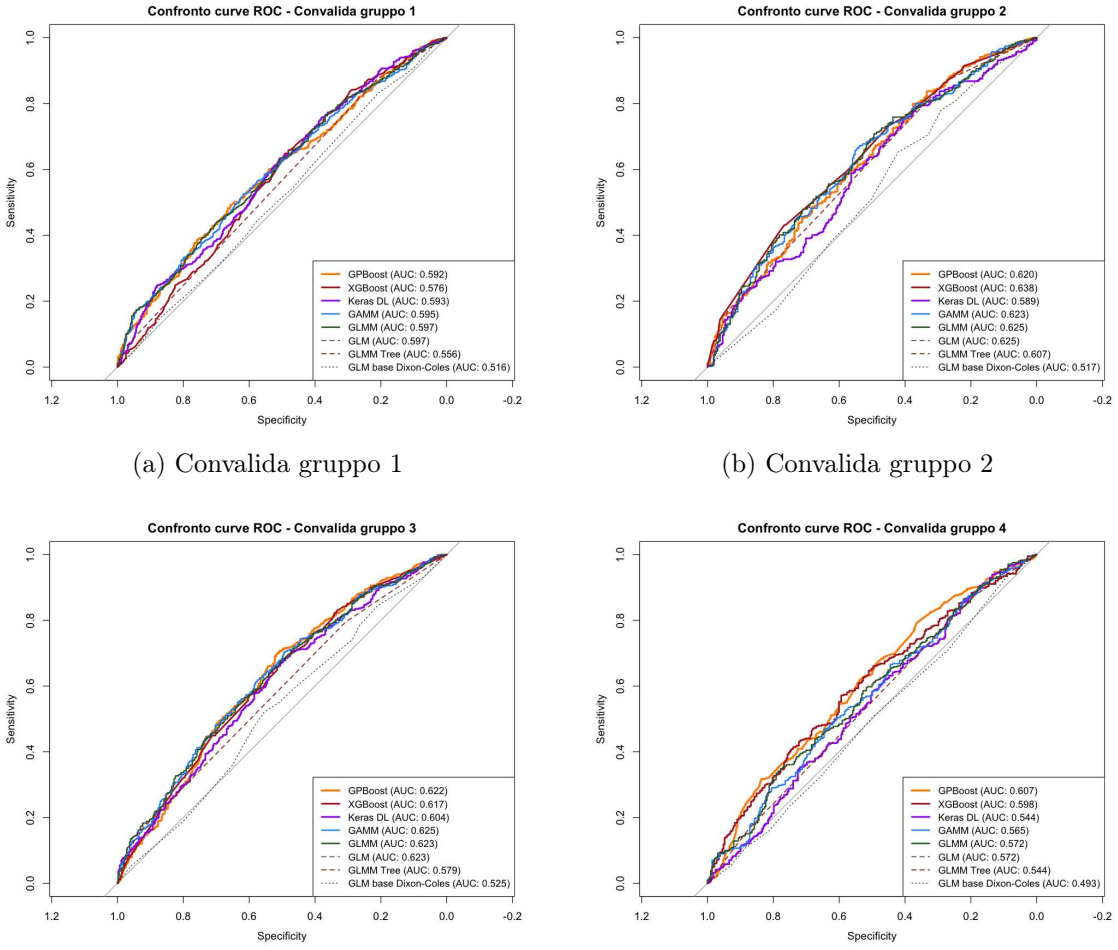
Famiglia e modello	AUC	F1	MCC
Modelli parametrici di riferimento			
GLM (solo con probabilità storiche (Dixon e Coles 1997))	0,5226	0,5582	-0,0184
GLM (*baseline* completamente indipendente)	0,5973	0,6285	0,1712
GLMM (*baseline* completamente gerarchica)	0,5973	0,6285	0,1712
Modello additivo non lineare			
GAMM (gerarchica non lineare)	0,5970	0,6148	0,1648
Modelli ad albero interpretabili			
GLM Tree (singolo, indipendente)	0,5691	0,6984	0,1373
GLMM Tree (singolo, gerarchico)	0,5691	0,6984	0,1373
Modelli di tipo *boosting*			
XGBoost (*boosting*, indipendente)	0,6043	0,6123	0,1811
GPBoost (*boosting*, gerarchico)	0,6048	0,6686	0,1805
Deep learning			
Keras (reti neurali profonde)	0,5994	0,6665	0,1724

In sintesi, i risultati mostrano che il GLM univariato semplice (solo storico) ha raggiunto un valore di AUC medio di 0,5226, leggermente superiore a quello della classificazione casuale. Tuttavia, aggiungendo le variabili contestuali multivariate (affaticamento, pressione competitiva) e le variabili spaziali (geometria dei primi tre tiri), il GLM multivariato è migliorato in modo sostanziale, raggiungendo un'AUC media di 0,5973.

Sebbene sia stata effettuata la transizione da una struttura indipendente a una gerarchica tramite il GLMM (*full hierarchical baseline*), non si è registrato alcun miglioramento predittivo in termini di AUC (0,5973). Questi risultati suggeriscono che le intercette casuali aggiuntive a livello di partita non catturano una quantità sufficiente di varianza marginale da causare una variazione significativa nell'area sotto la curva ROC.

Analogamente, anche l'allentamento dell'ipotesi di linearità utilizzando *spline* nel modello GAMM (*non-linear hierarchical*) non ha prodotto alcun miglioramento nell'AUC (0,5970). Nel complesso, questi risultati indicano che l'incapacità dei modelli tradizionali di superare la soglia di AUC di 0,60 è dovuta principalmente alla loro incapacità di catturare interazioni multiple (tattiche) complesse tra le variabili, piuttosto che alla non linearità dei singoli predittori.

Al fine di mostrare visivamente le prestazioni dei modelli in vari periodi di tempo, la Figura 5.1 illustra le curve ROC stimate per ciascun gruppo durante la convalida *walk forward*.



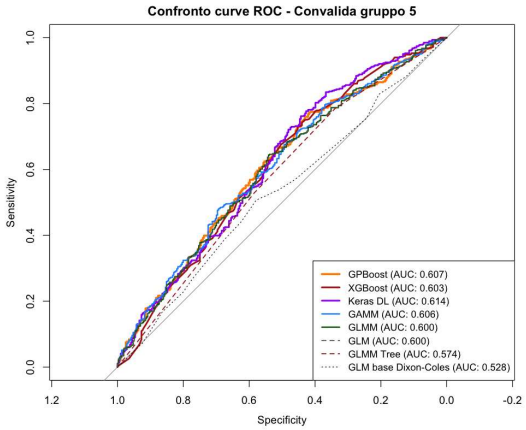
(a) Convalida gruppo 1

(b) Convalida gruppo 2

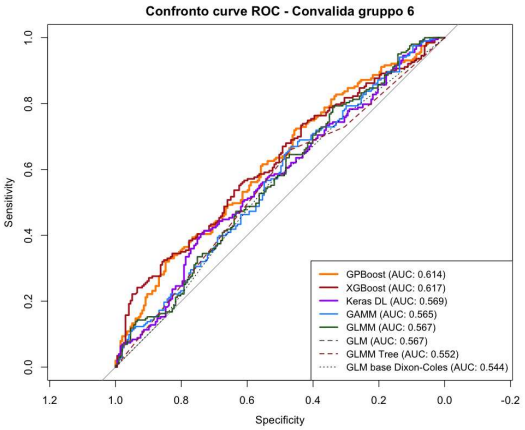
(c) Convalida gruppo 3

(d) Convalida gruppo 4

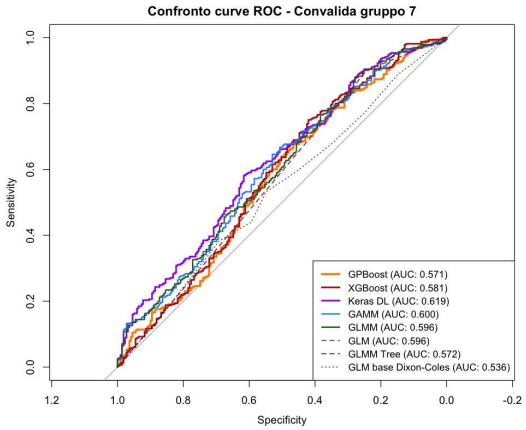
Figura 5.1 (Continua nella pagina successiva...)



(e) Convalida gruppo 5



(f) Convalida gruppo 6



(g) Convalida gruppo 7

Figura 5.1: Confronto delle curve ROC per i diversi algoritmi nei 7 fold di validazione *walk-forward*.

5.1 Risultati dei modelli ad albero singolo

I modelli basati sul partizionamento ricorsivo (GLM Tree e GLMM Tree) hanno ottenuto risultati peggiori rispetto alle controparti parametriche complete, raggiungendo un valore medio dell'indice AUC di 0,5691. Questo calo delle prestazioni è teoricamente attribuito all'instabilità intrinseca di un singolo albero decisionale.

A differenza dei metodi di regolazione dei CART tradizionali, per evitare il sovradattamento l'algoritmo MOB esegue test di fluttuazione, ma il singolo albero ha difficoltà a suddividere l'iperspazio del tennis senza perdere significatività statistica nei nodi terminali. Questo avviene perché il rapporto segnale/rumore tende ad essere molto debole. Pertanto, il modello basato su un singolo albero produce una stima delle probabilità meno flessibile e quindi meno discriminante rispetto al GLM.

5.2 Successo predittivo: la potenza dei metodi ensemble

I risultati della famiglia di modelli di *gradient boosting* confermano la supremazia, seppure in maniera minima, dei metodi *ensemble* per i dati sportivi tabulari. In particolare, XGBoost ha superato la barriera di 0,60 e ha raggiunto un valore AUC di 0,6043. Come accennato in precedenza, questo aumento è teoricamente giustificato dalla natura stessa del modello.

La sequenza di alberi poco profondi, ciascuno costruito sui residui dell'iterazione precedente, consente al modello di apprendere automaticamente le interazioni non lineari più forti tra le variabili esplicative (ad esempio, l'efficacia di un servizio largo può dipendere dallo stato di affaticamento dell'avversario e dalla pressione del tabellone). Queste interazioni sono troppo complesse per essere specificate manualmente in un GLM o GAMM.

Il modello che ha ottenuto i valori di AUC più elevati in assoluto è stato il *Gaussian process boosting* (GPBoost) (con indice AUC di 0,6048). Questo è riuscito a combinare i vantaggi dell'apprendimento non lineare (*boosting*) con la corretta modellazione della struttura di correlazione gerarchica dei dati.

Ottimizzando congiuntamente gli alberi decisionali e la matrice di covarianza degli effetti casuali (*match*, *set* e *game*), il modello ha separato efficacemente il segnale tattico puro dalla componente di varianza specifica del singolo incontro, ottenendo il modello più robusto per la previsione dei risultati dei punti.

Tuttavia, è evidente come il miglioramento predittivo di GPBoost rispetto a XGBoost sia estremamente marginale (inferiore allo 0,05%). Questo suggerisce che l'inclusione degli effetti casuali non apporti un contributo sostanziale: le specifiche dinamiche annidate relative a *match*, *set* e *game* sembrano influire in modo minimo e quasi nullo sull'esito del singolo punto. Va ricordato, però, che la modellazione del singolo scambio nel tennis è intrinsecamente dominata da un elevatissimo rumore stocastico; in tale contesto, anche un lieve incremento della capacità discriminante può essere considerato un successo degno di nota.

Osservando i risultati, emerge inoltre che anche la differenza di prestazioni rispetto al tradizionale modello parametrico di base (GLM) risulta contenuta. In linea di principio, si potrebbe valutare un compromesso metodologico: sacrificare una modesta frazione di potere predittivo a favore della totale interpretabilità offerta dal GLM. Tuttavia, quest'ultimo presenta un limite strutturale critico per gli scopi di questa ricerca. Per catturare l'effetto congiunto di diverse scelte tattiche, il GLM richiederebbe l'inserimento manuale e a priori di tutti i termini di interazione nella formula matematica. Dato l'elevato numero di variabili in gioco, l'esplorazione e la specificazione manuale di tutte le possibili interazioni significative risulta impraticabile. Al contrario, la struttura basata sul partizionamento ricorsivo (propria di XGBoost e GPBoost) è in grado di apprendere e mappare automaticamente queste complesse interazioni non lineari.

A maggior ragione, l'interpretazione diretta dei coefficienti marginali di un modello

parametrico su questo specifico set di dati risulterebbe insidiosa. Avendo applicato un filtro rigoroso per isolare esclusivamente gli scambi con almeno tre colpi validi, la stima dell'efficacia assoluta di un singolo colpo è soggetta a distorsione. Un coefficiente isolato si limiterebbe a indicare genericamente, a prescindere dal contesto, quanto sia conveniente servire in una certa direzione o tirare un terzo colpo incrociato. Tuttavia, per gli scopi di questa tesi, l'efficacia marginale assoluta non possiede alcun potere informativo: l'obiettivo è costruire vere e proprie tattiche e *pattern* di gioco. Di conseguenza, è fondamentale valutare dove direzionare il terzo colpo *condizionatamente* alla direzione del servizio precedente. L'efficacia di tali combinazioni, inoltre, potrebbe subire variazioni drastiche in base ad altre variabili di confondimento, come la superficie di gioco o l'inerzia psicologica del momento.

Per rispondere a queste domande multidimensionali, il ricorso a simulazioni di scenari (come verrà affrontato nel Capitolo 7) si rivela lo strumento analitico più adeguato. Per questi motivi, si decide di eleggere GPBoost come modello finale di riferimento per la previsione del singolo punto e per le successive simulazioni. Sebbene restituisca risultati empiricamente analoghi a XGBoost, la scelta ricade sull'algoritmo che ha dimostrato la massima capacità predittiva assoluta sul campione in esame (GPBoost).

Pur riconoscendo le suddette limitazioni metodologiche e concettuali relative all'approccio parametrico puro, si ritiene comunque propedeutico e di interesse esplorativo riportare le stime dei coefficienti del modello GLM. Tali stime saranno accompagnate di seguito da un grafico degli intervalli di confidenza (una rappresentazione visiva degli effetti marginali), al fine di offrire al lettore una prima lettura quantitativa e intuitiva delle dinamiche di base prima di addentrarsi nella complessità simulativa.

Tabella 5.2: Stima dei coefficienti del modello GLM (baseline indipendente) sull'ultima finestra temporale di convalida (*Fold* 7). Gli intervalli di confidenza al 95% evidenziano l'esplosione della varianza per alcuni livelli categoriali rari.

Termine	**Stima**	**Int. Confidenza 95%**	**P-Value**	
(Intercept)	$4,553$	$[-11,243;20,349]$	$0,572$	
`SurfaceGrass`	$0,090$	$[-1,007;1,186]$	$0,873$	
`SurfaceHard`	$-0,410$	$[-0,813;-0,008]$	$0,046$	*
`tournament_importance`	$0,144$	$[-0,106;0,394]$	$0,258$	
`round_numeric`	$-0,261$	$[-0,657;0,136]$	$0,197$	
`score_status0`	$0,006$	$[-0,332;0,344]$	$0,974$	
`score_status1`	$-0,059$	$[-0,470;0,352]$	$0,779$	
`importanza`	$-0,003$	$[-0,762;0,756]$	$0,993$	
`is_new_balls1`	$0,133$	$[-0,153;0,419]$	$0,361$	
`is_deciding_set1`	$-0,426$	$[-1,089;0,237]$	$0,208$	
`serve_sideDeuce`	$0,006$	$[-0,258;0,271]$	$0,962$	
`momentum_focal`	$0,093$	$[-0,774;0,960]$	$0,834$	
`opp_handR`	$-0,325$	$[-1,116;0,466]$	$0,420$	
`opp_serve_eff`	$-1,629$	$[-9,707;6,449]$	$0,693$	
`opp_return_qual`	$-0,749$	$[-9,069;7,571]$	$0,860$	
`opp_ace_rate`	$-0,087$	$[-11,177;11,003]$	$0,988$	
`opp_aggression`	$-1,492$	$[-5,077;2,092]$	$0,414$	
`serve_num`	$0,036$	$[-0,221;0,294]$	$0,783$	
`serve_dir`	$-0,130$	$[-0,265;0,005]$	$0,060$	.
`is_serve_volley`	$0,467$	$[-2,315;3,249]$	$0,742$	
`ret_typef`	$0,030$	$[-0,240;0,300]$	$0,827$	
`ret_typel`	$13,317$	$[-1985,121;2011,755]$	$0,990$	
`ret_typem`	$13,625$	$[-1105,664;1132,913]$	$0,981$	
`ret_typer`	$0,387$	$[-0,262;1,036]$	$0,243$	
`ret_types`	$0,738$	$[0,081;1,396]$	$0,028$	*
`ret_dir2`	$0,157$	$[-0,248;0,561]$	$0,448$	
`ret_dir3`	$-0,145$	$[-0,634;0,343]$	$0,560$	
`shot3_typef`	$-0,027$	$[-0,400;0,346]$	$0,889$	
`shot3_typeh`	$-0,945$	$[-4,376;2,485]$	$0,589$	
`shot3_typei`	$-16,246$	$[-2028,913;1996,422]$	$0,987$	
`shot3_typej`	$0,554$	$[-2,657;3,766]$	$0,735$	
`shot3_typem`	$-15,534$	$[-2868,113;2837,045]$	$0,991$	
`shot3_typeo`	$1,404$	$[-1,393;4,201]$	$0,325$	
`shot3_typer`	$-0,947$	$[-1,855;-0,039]$	$0,041$	*
`shot3_types`	$0,041$	$[-0,621;0,702]$	$0,904$	
`shot3_typeu`	$0,511$	$[-1,144;2,167]$	$0,545$	
`shot3_typev`	$-1,155$	$[-4,132;1,821]$	$0,447$	
`shot3_typey`	$-1,469$	$[-3,098;0,160]$	$0,077$	.
`shot3_typez`	$-0,322$	$[-3,302;2,658]$	$0,832$	
`shot3_dir2`	$-0,881$	$[-1,251;-0,510]$	$< 0,001$	***
`shot3_dir3`	$-0,040$	$[-0,326;0,246]$	$0,785$	
`Player_Baseline_Prob`	$-1,174$	$[-19,761;17,414]$	$0,902$	

*Nota: Livelli di significatività: . $p < 0,1$; * $p < 0,05$; ** $p < 0,01$; *** $p < 0,001$.*

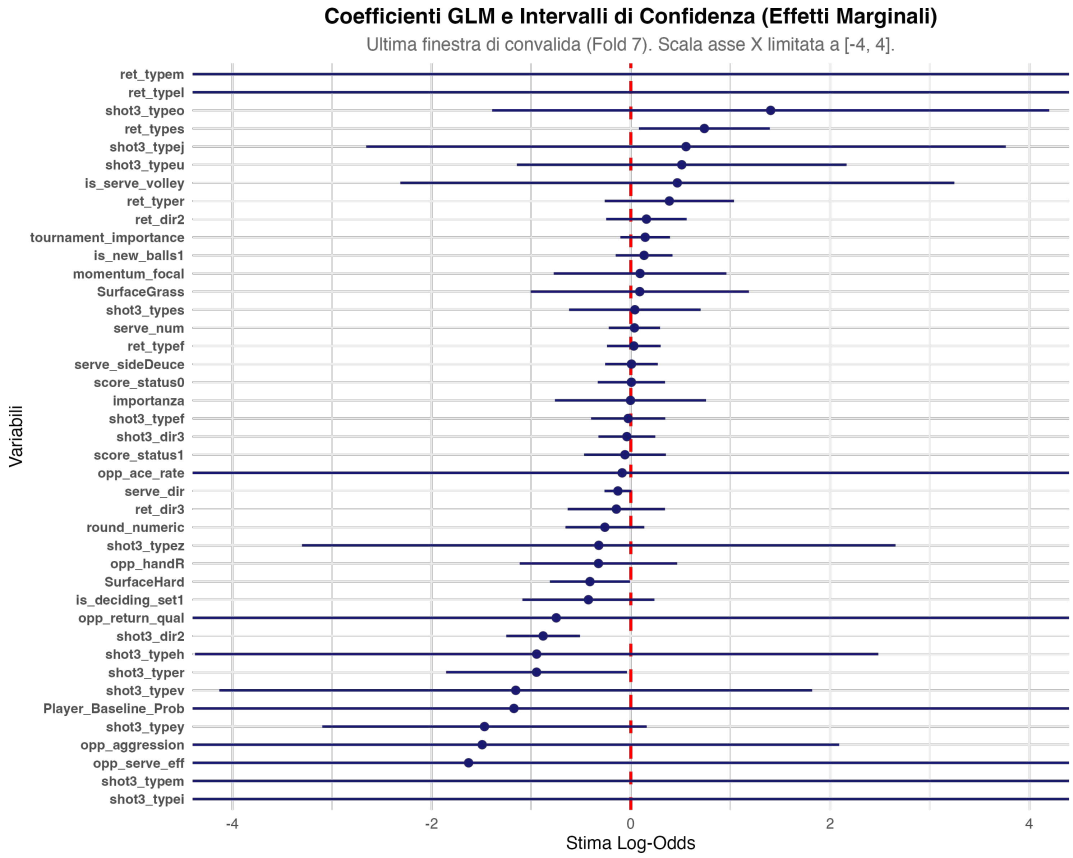


Figura 5.2: Rappresentazione visiva dei coefficienti stimati dal modello GLM (*baseline* indipendente) e dei relativi intervalli di confidenza al 95% sull'ultima finestra temporale di convalida (*Fold* 7). Il grafico illustra la direzione e l'ampiezza degli effetti marginali di ciascuna variabile sull'esito del punto. (Nota: l'asse delle ascisse è stato troncato all'intervallo $[-4, 4]$ per consentire una corretta visualizzazione e isolare le variabili categoriali caratterizzate da eccessiva varianza).

5.3 Risultati della metodologia della rete neurale e discussione sull'interpretabilità

Il metodo di *deep learning* è stato implementato utilizzando una rete neurale *feedforward* (a propagazione in avanti) utilizzando Keras e ha raggiunto un'AUC di 0,5994. Sebbene ciò rappresenti una buona prestazione in questo contesto, l'indice AUC inferiore rispetto alle metodologie di *boosting* è coerente con la letteratura sui dati tabulari.

Le reti neurali in genere richiedono grandi quantità di dati per produrre buoni risultati e soffrono di problemi di interpretabilità legati alla loro complessità e alla tendenza a sovra-adattare i dati di stima in presenza di forte rumore stocastico.

Pertanto, sebbene la regolarizzazione *dropout* di Keras (spiegata nel capitolo 4) sia stata applicata per prevenire il sovradattamento, le scarse prestazioni del modello di *deep*

learning in questo esperimento potrebbero essere dovute al sottocampionamento effettuato per filtrare i dati di uno specifico giocatore al servizio e righe corrispondenti a scambi di almeno 3 colpi, che ha sensibilmente ridotto la dimensione dell'insieme di stima. In questo caso ci si aspettava questo risultato e il modello è stato adattato con il solo scopo di confrontarlo in modo equo con gli altri e confermare quanto ci dice la letteratura a riguardo. Tuttavia, un possibile sviluppo può essere quello di provare a creare un'architettura per modellare l'intero *dataset* di tutti i giocatori tramite modelli di *deep learning* e vedere se la capacità predittiva a livello di singolo punto migliora rispetto a quanto ottenuto fino ad ora.

Infine, si è valutato di costruire un'analisi grafica SHAP (Lundberg e Lee 2017) per commentare i risultati, ma poiché in fase di analisi preliminare sono state trasformate le direzioni e i tipi di colpo (che spiegano gli schemi di gioco) in variabili indicatrici, il calcolo dei contributi SHAP marginali per una singola riga porterebbe a un'interpretazione complessa e frammentata. Basti pensare che le variabili relative al tipo di colpo hanno circa 20 modalità. Inoltre, la cosa interessante per questo studio non è dire quali colpi favoriscono maggiormente la vittoria del punto, bensì quali combinazioni (e quindi interazioni) di colpi portano vantaggio, ossia gli schemi tattici vincenti.

Pertanto, l'obiettivo di decifrare il modello *black-box* (GPBoost) sarà perseguito nel capitolo 6 utilizzando una metodologia di simulazione tattica che genererà linee guida strategiche applicabili sul campo basate sulle previsioni a livello di popolazione.

Capitolo 6

Simulazione tattica e inferenza predittiva

6.1 Metodologia per la simulazione di scenari

Lo scopo di questo capitolo è convertire il complesso modello predittivo ottimale (GPBoost) sviluppato nel capitolo 5 in raccomandazioni strategiche attuabili e pratiche che possano essere interpretate dagli allenatori. Poiché il modello produce una superficie di probabilità altamente non lineare, non è possibile analizzare ogni singolo coefficiente. Di conseguenza, è stato impiegato un metodo di analisi basato su scenari. Stabilendo le variabili contestuali per una condizione di partita specifica e fissandole a quei valori, è possibile produrre un insieme di dati sintetico in cui variano solo le opzioni tattiche del giocatore che serve. Il modello viene interrogato sulla griglia delle combinazioni delle diverse opzioni del giocatore che serve. A sua volta, dal modello è possibile ottenere le probabilità condizionate di vincere il punto. Questo tipo di analisi consente al ricercatore di determinare l'effetto marginale dello schema *serve + 1* (il primo colpo eseguito dal giocatore che serve dopo la risposta dell'avversario) mantenendo costanti gli effetti dei fattori di contesto.

6.2 Definizione del contesto della partita: Tsitsipas vs. Zverev

Per stabilire un contesto realistico e di alto livello per lo studio, è stata simulata una partita ipotetica tra Stefanos Tsitsipas (al servizio) e Alexander Zverev (in risposta, giocatore destrorso). L'insieme delle variabili contestuali è stato stabilito per rappresentare accuratamente un momento di media pressione competitiva in un torneo *Masters 1000* su campi in cemento, ovvero simile alle condizioni del torneo di Miami. Nello specifico, la simulazione è stata condotta nei quarti di finale durante un set decisivo. Tsitsipas stava servendo dal campo Ad (il lato sinistro del campo) con Zverev che affrontava un *break point*, anche se il *break point* non costituiva un *match point*. Per concentrarsi esclusivamente sugli aspetti

tattici della partita, lo slancio psicologico è stato impostato su un livello neutro e si è ipotizzato l'uso di palline non nuove (cioè non i primi due game dopo un cambio di palline).

Le variabili storiche associate all'avversario necessarie per inserire il modello sono state determinate aggregando le statistiche di Zverev (ipotetico avversario) nei 365 giorni precedenti la data ipotetica della partita (in questo casa la data di riferimento è la data dei quarti di finale del torneo di Miami 2025). Queste variabili includevano la prestazione al servizio dell'avversario, la percentuale di *ace* concessi, la propensione all'aggressività dell'avversario e la qualità complessiva della risposta. Inoltre, la probabilità di base pre-partita del giocatore (*Player_Baseline_Prob*) è stata inserita nel modello recuperando i coefficienti precedentemente stimati dal modello parametrico di Dixon e Coles (Dixon e Coles 1997). Ciò ha garantito che la stima tattica intra-partita fosse correttamente ancorata alla forza latente complessiva dei due giocatori.

6.3 Troncamento dello spazio campionario e limiti interpretativi

Prima di condurre l'analisi dei modelli tattici, è necessario identificare un limite metodologico fondamentale relativo alla natura dei dati esaminati. I dati esaminati sono stati filtrati per includere solo gli scambi con un minimo di tre colpi (lunghezza dello scambio ≥ 3). Da un punto di vista statistico, questo filtro rappresenta un troncamento dello spazio campionario. Tutti i punti risolti in uno o due colpi sono stati eliminati dallo spazio campionario. Storicamente, i servizi al centro (lungo la T) hanno rappresentato la più alta probabilità di risultare in un *ace* o in un servizio vincente nel tennis. Di conseguenza, filtrando i dati per includere solo gli scambi con una lunghezza maggiore o uguale a tre, stiamo essenzialmente rimuovendo dal campione i servizi T più efficaci e includendo solo quelli che l'avversario è riuscito a neutralizzare e a restituire in gioco. Ciò si traduce in un fenomeno ben noto denominato distorsione di sopravvivenza. Se si confrontassero le probabilità marginali di vittoria tra le diverse direzioni in cui vengono eseguiti i servizi, il servizio lungo la T risulterebbe molto sottorappresentato e apparirebbe come l'opzione peggiore, poiché viene valutato solo in base ai suoi "fallimenti", ovvero quando l'avversario ha restituito il servizio. Di conseguenza, piuttosto che valutare l'efficacia assoluta di un servizio largo, sul corpo o sulla T, l'analisi deve limitarsi a determinare le probabilità condizionate: dato che un servizio è stato eseguito in una particolare direzione e dato che lo scambio è progredito fino al terzo colpo, qual è la direzione migliore per eseguire il colpo successivo del battitore? Tuttavia, dal punto di vista metodologico, l'analisi dello schema tattico *Serve + 1* non pone problemi di validità: poiché anche nella stima dei modelli predittivi sono stati selezionati esattamente questi dati, la popolazione di interesse e la popolazione di stima essenzialmente coincidono.

6.4 Analisi dei modelli nello schema *serve + 1*

Sulla base del contesto di partita stabilito, il modello ha stimato le probabilità condizionate di Tsitsipas di vincere il punto al variare della direzione del servizio e della successiva scelta spaziale del terzo colpo (a sinistra, al centro o a destra, sempre dal punto di vista di Tsitsipas che affronta la metà campo dell'avversario). Gli scenari ipotizzavano le risposte più comuni di Zverev in base alla traiettoria del servizio. Prima di procedere con l'analisi dettagliata, la mappa di calore presentata in Figura 6.1 fornisce una visione d'insieme delle probabilità stimate per tutte le combinazioni.

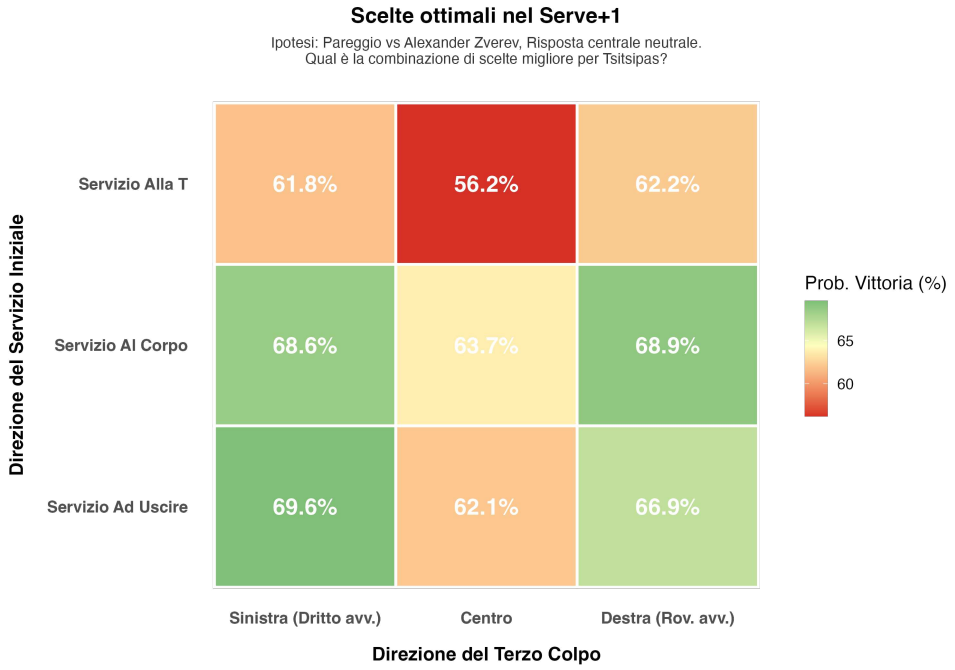
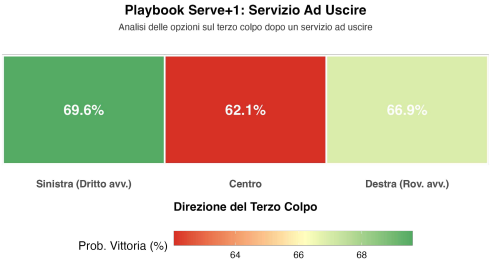


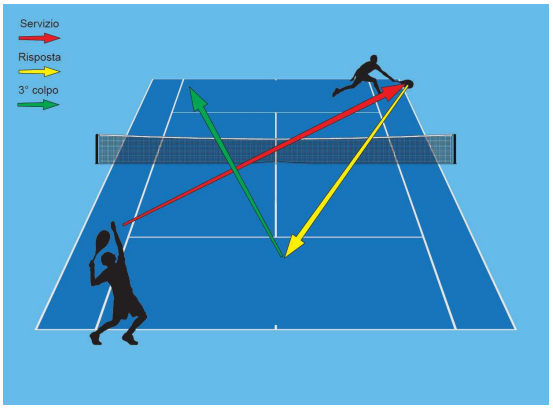
Figura 6.1: Mappa di calore riassuntiva delle probabilità condizionate di vittoria in funzione della direzione del servizio e del terzo colpo.

6.4.1 Scenario A: servizio largo

Quando Tsitsipas serve largo dal campo Ad (lato sinistro), costringe Zverev a eseguire una risposta di rovescio. Il modello ipotizza che Zverev risponderà con un rovescio centrale profondo. In queste circostanze spaziali, come illustrato nella figura 6.2 che affianca le probabilità alla rappresentazione sul campo, la stima predittiva ha indicato che la decisione ottimale per Tsitsipas per massimizzare la probabilità di vincere sul terzo colpo è quella di dirigere la palla a sinistra. In molti casi, questa decisione tattica si traduce nell'esecuzione di un diritto *inside-out* che cerca di spiazzare Zverev sul suo lato di diritto, o nel tentativo di costringere l'avversario fuori dal campo enfatizzando gli angoli.



(a) Probabilità terzo colpo

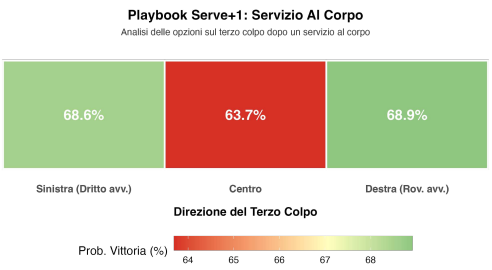


(b) Rappresentazione spaziale

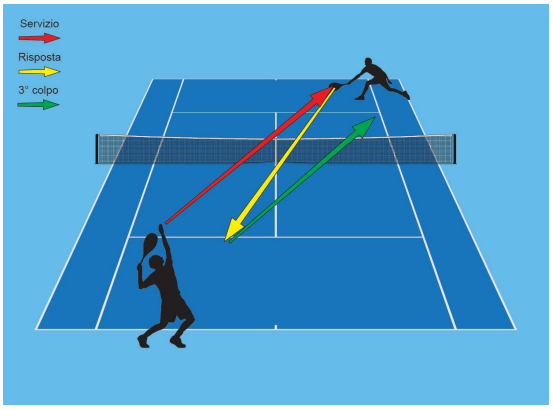
Figura 6.2: Analisi dello schema a seguito di un servizio largo e risposta centrale.

6.4.2 Scenario B: servizio sul corpo

Quando Tsitsipas indirizza il servizio sul corpo dell'avversario, riduce lo spazio di *swing* dell'avversario. Supponendo che Zverev sia in grado di tornare al centro del campo con un rovescio, il risultato del modello (riportato nella figura 6.3) ha suggerito un cambiamento radicale di direzione sul terzo colpo. In questo caso, la probabilità condizionata di vittoria è massimizzata se Tsitsipas esegue il colpo a destra (lato rovescio di Zverev). Dirigendo il servizio verso il corpo, il battitore non crea spazio aggiuntivo nel campo per l'attacco dell'avversario; giocando a destra, Tsitsipas crea un difficile disorientamento posturale sul lato più scomodo per l'avversario.



(a) Probabilità terzo colpo



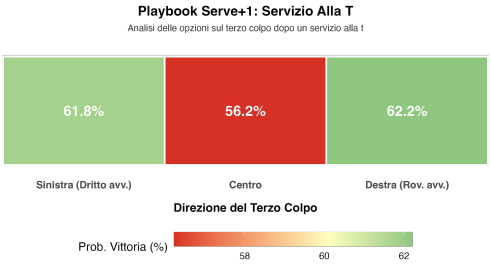
(b) Rappresentazione spaziale

Figura 6.3: Analisi dello schema a seguito di un servizio sul corpo e risposta centrale.

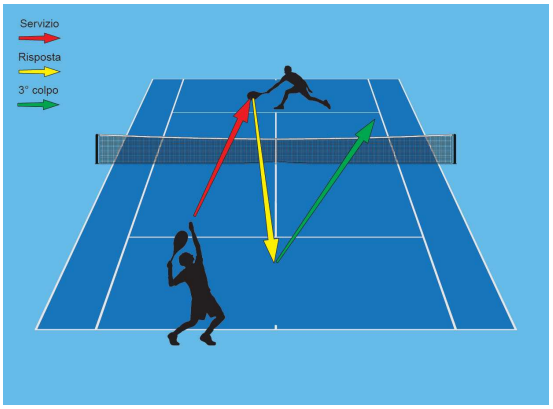
6.4.3 Scenario C: servizio al centro (lungo la linea T)

Quando Tsitsipas serve al centro (lungo la linea T) dal campo Ad, costringe Zverev a rispondere con un diritto. Supponendo che Zverev risponda con un diritto centrale, il

modello ha nuovamente identificato (si veda la figura 6.4) il lato destro del campo (il rovescio dell'avversario) come zona di destinazione ottimale per il terzo colpo di Tsitsipas. Quando Tsitsipas serve lungo la T, costringe Zverev a spostarsi verso il centro. Quando Tsitsipas cambia rapidamente ritmo verso destra, costringe Zverev a cambiare rapidamente direzione per difendere l'angolo esposto, massimizzando così il rendimento tattico dello schema.



(a) Probabilità terzo colpo



(b) Rappresentazione spaziale

Figura 6.4: Analisi dello schema a seguito di un servizio al centro (lungo la linea T) e risposta centrale.

6.5 Riepilogo delle direttive strategiche

Nel complesso, l'applicazione della simulazione predittiva ha dimostrato che esistono schemi tattici che possono essere utilizzati per gestire i *break point*. A differenza delle ricerche precedenti che si basavano esclusivamente sull'intuizione, l'approccio statistico ha dimostrato che la direzione del terzo colpo dipende dalla direzione del primo colpo (servizio) e che il terzo colpo deve essere adattato geograficamente alla direzione del primo colpo. In sostanza, per massimizzare la probabilità di salvare un *break point* contro un ribattitore con le caratteristiche di Zverev, Tsitsipas dovrebbe utilizzare uno schema *cross-court* (terzo colpo a sinistra) dopo un servizio largo, mentre dovrebbe scegliere di mirare al rovescio dell'avversario (terzo colpo a destra) se decide di utilizzare un servizio sul corpo o un servizio lungo la linea centrale.

Conclusioni

Sintesi della ricerca

L'obiettivo principale di questa tesi era quello di creare un quadro scientifico per misurare l'efficacia delle tattiche impiegate dai professionisti maschi nel tennis ATP. Per evitare i limiti della statistica descrittiva, come la semplice segnalazione della frequenza degli stili di gioco dei giocatori, questa ricerca si è concentrata sulla creazione di un modello per stimare la probabilità marginale effettiva di successo di un giocatore a livello di singolo "punto". Il successo del giocatore dipende da molteplici fattori che influenzano la sua capacità di vincere i singoli scambi (affaticamento, pressione del punteggio e livello di abilità nascosto di ciascun giocatore).

Per determinare se è possibile creare un modello in grado di prevedere con precisione la capacità di un giocatore di avere successo quando tenta di conquistare un "punto", sono stati applicati metodi di apprendimento statistico di tipo supervisionato. Questi includevano modelli parametrici semplici (GLM e GLMM) e architetture di reti neurali profonde. Tutti i modelli sono stati verificati e convalidati utilizzando un metodo di convalida *walk-forward* per misurare l'accuratezza predittiva di questi modelli, tenendo conto delle dipendenze sequenziali e della dipendenza temporale dei dati.

Considerazioni finali dei risultati dell'analisi

Le prestazioni complessive dei modelli hanno identificato che il metodo *gradient boosting* era superiore a tutti gli altri modelli in termini di accuratezza predittiva. Il modello con le migliori prestazioni, GPBoost (Sigrist 2022), ha ottenuto l'AUC più alto, pari a 0,6048, grazie alla sua flessibilità sia nell'includere la stima non parametrica delle interazioni complesse tra le variabili tattiche (utilizzando la combinazione di alberi decisionali) sia nel fornire una buona rappresentazione della gerarchia dei dati (incorporando effetti casuali nidificati per rappresentare la partita, il *set* e il gioco).

Per convertire le stime prodotte dal modello complesso in consigli strategici attuabili, è stata sviluppata una metodologia di inferenza basata sulla simulazione di scenari. Utilizzando le variabili contestuali per identificare un "punto" della partita in cui la pressione è alta

ma non massima (quando un giocatore affronta un *break point* contro Stefanos Tsitsipas e Alexander Zverev), l'analisi della tattica "*Serve + 1*" dimostra chiaramente che la posizione del terzo colpo in uno scambio ha un alto grado di dipendenza condizionata dalla posizione del servizio. Nello specifico, il modello stima che se un giocatore utilizza un servizio ampio, il terzo colpo dovrebbe essere piazzato sul lato opposto (*cross court*) per aumentare le probabilità di successo, mentre se un giocatore utilizza un servizio al corpo o lungo la linea, il terzo colpo sarà più efficace se attacca il lato più svantaggiato e cogliendo in contropiede l'avversario (sul rovescio).

Limiti dello studio

Sebbene la solidità dei risultati di questo studio sia evidente, il quadro scientifico dello studio presenta alcuni limiti. Il limite statistico più importante deriva dai criteri di inclusione del campione. Poiché nel campione sono stati inclusi solo scambi di almeno tre colpi (lunghezza dello scambio > 3), lo spazio campionario è stato troncato, con conseguente *bias* di sopravvivenza. Infatti, dal momento che tutti gli *ace* e i servizi non restituiti sono stati esclusi dal campione, l'efficacia assoluta dei primi servizi (in particolare quelli diretti lungo la T) è intrinsecamente sottostimata, limitando così la validità dell'interpretazione strategica esclusivamente all'ambito delle probabilità condizionate una volta iniziato lo scambio.

Una seconda preoccupazione importante riguarda le prestazioni predittive relativamente modeste delle reti neurali. Le reti neurali richiedono in genere un campione di dimensioni molto grandi per stimare efficacemente il modello senza incorrere in un sovradattamento. A causa del sottocampionamento effettuato per isolare esclusivamente i giochi di servizio di un singolo giocatore, la dimensione degli insiemi di stima si è ridotta di molto e, conseguentemente, l'efficacia delle reti neurali è stata influenzata negativamente rispetto ai modelli basati sulla partizione ricorsiva dello spazio, noti per essere molto più stabili quando si tratta di piccoli *set* di dati tabulari con un basso rapporto segnale/rumore.

Possibili sviluppi futuri della ricerca

Sia i limiti emersi che le prove empiriche raccolte dallo studio attuale suggeriscono diverse direzioni per la ricerca futura. Dal "punto" di vista statistico, uno dei passi successivi può essere quello di sviluppare un modello a livello di popolazione sull'intero *set* di dati del *Match Charting Project*, comprese le traiettorie di tutti i giocatori che competono nel *tour*. Ciò consentirebbe l'estrazione delle componenti generali sottostanti del gioco e permetterebbe la calibrazione delle stime alle specificità dei singoli giocatori di tennis tramite, ad esempio, inferenza bayesiana o metodologie di *borrowing of information*. Inoltre,

sarebbe interessante valutare e confrontare l'accuratezza dell'architettura attuale creata applicandola a dati Hawk-Eye di alta qualità e precisione.

Infine, dal punto di vista della teoria delle decisioni, le stime di probabilità condizionata prodotte dal modello GPBoost potrebbero essere formalizzate nei teoremi della teoria dei giochi. Poiché nel tennis il battitore e il ricevitore sono concorrenti in un gioco non cooperativo a somma zero, l'uso delle probabilità condizionate stimate nel capitolo 6 consentirebbe di determinare l'equilibrio (o gli equilibri) di Nash per le strategie miste del gioco. Ciò consentirebbe non solo di identificare le azioni con la massima utilità attesa, ma anche di quantificare le distribuzioni di probabilità ottimali necessarie per selezionare in modo casuale la direzione del servizio e della risposta. Ciò eliminerebbe la capacità dell'avversario di anticipare gli schemi di servizio e di risposta e porterebbe l'analisi sportiva a un nuovo livello di profondità analitica.

Appendice A

Pre-elaborazione avanzata e analisi sintattica dei dati

Questo capitolo illustra in dettaglio le architetture informatiche e le strategie logiche adottate per ripulire la base dati grezza estratta dal *Match Charting Project* (MCP). L'origine *crowdsourced* (collaborativa) dei dati comporta fisiologicamente una dose di rumore sintattico e anomalie strutturali che avrebbero compromesso la successiva fase di ingegneria delle variabili e, conseguentemente, l'accuratezza dei modelli predittivi. L'intera elaborazione è stata condotta in ambiente R, limitando l'analisi ai dati aggiornati al 14 giugno 2025.

A.1 Integrità strutturale e manutenzione dei dati

Prima di procedere all'analisi complessa delle sequenze dei colpi, è stata garantita l'integrità strutturale del database tramite la funzione `check_data_integrity`. Tale procedura opera una scansione granulare del materiale, controllando la corretta sequenzialità di partite, *game* e *set*, isolando le anomalie e provvedendo alla correzione manuale o semi-automatica dei *record* compromessi. Di seguito si riportano le casistiche più rilevanti che hanno richiesto un intervento diretto:

- **Frammentazione e sfasamento dei record:** Nello storico dei match, i *record* n. 447 e 448 risultavano frammentati e sovrapposti, pur riferendosi al medesimo incontro. Presentavano incongruenze evidenti, come la stringa anomala "R R" in luogo dei nomi dei giocatori e una compilazione parziale dei metadati. I dati sono stati ricongiunti e sintetizzati in un'unica osservazione coerente.
- **Traslazione (*shift*) delle variabili:** Nel match *Griekspoor-Cobolli* (Coppa Davis 2024), l'omissione dell'annotazione relativa alla superficie di gioco ha causato uno

slittamento orizzontale di tutte le colonne successive. Il campo mancante ("hard") è stato imputato ripristinando il corretto allineamento tabellare.

- **Incompletezza di punteggio e metadati:** Nel match *Passaro-Kjaer* (Montemar Challenger 2024), i dati relativi ai *game* vinti dal secondo giocatore (`Gm2`) sono stati ricalcolati deterministicamente. Nel match *Alcaraz-Coria* (Rio de Janeiro 2020), è stato ripristinato il *flag* relativo alla disputa del *tie-break* (impostato su "True"), originariamente omesso.
- **Rimozione di duplicati e record danneggiati:** Sono state eliminate le doppie occorrenze dei match *Hurkacz-Kyrgios* (2022) e *Davidovich Fokina-Jaziri* (2023). Inoltre, durante il *parsing* è emerso che alcuni match presentavano errori sintattici in oltre il 50% dei punti tracciati; tali incontri sono stati rimossi dal dataset per evitare distorsioni statistiche (es. *Cobolli-Alcaraz*, *Alcaraz-Shapovalov*, *Munar-Cerundolo*).

A.2 Coerenza logica e sequenziale degli scambi

Risolta l'anagrafica, è stata implementata la funzione `logic_check` per verificare la coerenza interna alle singole sequenze di colpi. La natura manuale della trascrizione ha generato disomogeneità tra quanto registrato nel primo servizio (`1st`) e nel secondo (`2nd`), a cui si è posto rimedio analizzando le seguenti casistiche:

- **Punti di servizio "Fantasma":** In diverse occorrenze, nonostante il primo servizio risultasse valido, il trascrittore aveva compilato erroneamente anche il campo della seconda (es. *Struff-Auger Aliassime*, ATP Cup 2020). In altri casi, la prima risultava conclusa ma il campo della seconda conteneva la trascrizione errata del punto successivo (es. *Wawrinka-Djokovic*, Roma 2022). In tali ipotesi, la seconda è stata eliminata per ripristinare la corretta successione temporale.
- **Lapsus di trascrizione:** Frequenti i casi in cui lo scambio sulla prima risultava manomesso o incompleto, mentre la seconda conteneva la trascrizione integra e logicamente coerente. In circa quaranta circostanze l'automatismo ha sovrapposto lo scambio corretto della seconda sulla prima.
- **Sostituzione di marcatori vaghi:** Laddove la prima recasse solo il marcatore "S" (indicante la vittoria del servitore senza dettagli tattici), ma la seconda riportasse lo scambio completo, si è proceduto all'integrazione della sequenza dettagliata.

A.3 Analisi sintattica tramite espressioni regolari (*Regex*)

Il cuore del processo di pulizia è rappresentato dall'estrazione dell'informazione spaziale tramite *Regular Expressions*. La sintassi MCP richiede che ogni colpo sia definito da una

lettera indicativa del tipo (es. `f` per diritto, `b` per rovescio), seguita da un numero direzionale (1, 2, 3, 0) ed eventuali modificatori speciali.

Per normalizzare le stringhe e risolvere il disordine sintattico causato da errori di digitazione, è stata applicata una procedura di pulizia che ha rimosso gli spazi (`clean_spaces`) e riordinato i componenti della stringa.

A.3.1 Rimozione di caratteri illeciti e troncamento

La funzione `check_illegal_chars` ha permesso di allineare le serie contenenti artefatti digitali o caratteri non ammessi. Se una sequenza valida presentava un carattere terminatore (es. `*` per un vincente o `@` per un errore non forzato) seguito da rumore alfanumerico (es. *Mensik-Hurkacz*, Roma 2025), la stringa è stata troncata all'interruttore logico. Simboli come `)*` sono stati corretti in `0*`.

A.3.2 Gestione del marcatore Let (c)

Il marcatore `c` (nastro al servizio) deve trovarsi esclusivamente all'inizio della stringa. Tramite `remove_spurious_c`, sono state epurate le occorrenze disarmonizzate. Inoltre, i simboli speciali (`+`, `;`, `^`, `-`, `=`) sono stati riposizionati correttamente tra il tipo di colpo e la direzione. Ad esempio, stringhe come `4w^` sono state normalizzate in `4ŵ` (`fix_service_net_modifier`) e l'ordine tra tipo di errore e terminatore è stato corretto (es. da `@!` a `!@`). Tale intervento ha sanato automaticamente 550 sequenze.

A.3.3 Risoluzione delle ambiguità direzionali

In presenza di cifre direzionali doppie (due numeri consecutivi senza lettera interposta), l'algoritmo ha applicato una logica di sfasamento: se la cifra doppia era seguita da due lettere consecutive, il primo numero veniva associato al colpo successivo; in caso contrario, è stata conservata solo l'ultima cifra, interpretando il numero precedente come un errore di digitazione corretto istantaneamente dal trascrittore.

A.4 Impostazione strutturale delle coordinate mancanti

Per evitare l'introduzione di valori mancanti (`NA`) che avrebbero compromesso l'addestramento dei modelli (*data drop*), si è proceduto all'imputazione di marcatori neutri nelle coordinate vitali assenti:

1. **Direzione ignota (imputazione di "0"):** Circa 14.029 righe presentavano colpi senza specifica di traiettoria. Ad esse è stato apposto il carattere convenzionale `0`, previsto dalla sintassi MCP per indicare una direzione incerta.[1]

[1] Si sottolinea che il carattere "0" è regolarmente previsto dal protocollo MCP e non costituisce un artificio statistico introdotto in questa sede.

2. **Specie di errore mancante (imputazione di "e"):** In assenza di specifica tra il tipo di colpo e il terminatore (`#` o `@`), è stato inserito il carattere `e` (errore generico).

3. **Primo carattere non idoneo:** Se una sequenza iniziava con un carattere non ammesso per il servizio, il primo carattere è stato forzato a `0`.

Questo approccio garantisce due vantaggi matematici fondamentali: preserva la lunghezza logica della stringa, dato essenziale per il calcolo della durata dello scambio, e codifica l'assenza di informazione spaziale come una categoria discreta interpretabile dai modelli ad albero, senza indebolire la base informativa del dataset.

Appendice B

Codice R per la modellazione

In questa appendice è riportato il codice R integrale sviluppato per l'addestramento dei modelli predittivi all'interno dell'architettura di validazione *walk-forward*. Il codice include le fasi di *tuning* degli iperparametri e la valutazione comparativa tramite metrica AUC.

```r
# ============================================================
# FASE 4: KERAS (DEEP LEARNING CON TUNING)
# ============================================================
cat("\n--- FASE 4: Modellazione Keras (Sequenziale con Grid Search) ---\n")

risultati_keras <- data.frame()
keras_preds_list <- list() # Lista per salvare le previsioni Keras da usare poi nel plot ROC

for(i in 1:nrow(finestre_df)) {
  cat(paste0("\n[", format(Sys.time(), "%H:%M:%S"), "] START Fold ", i, " (Keras Deep Learning - 3 Colpi)\n"))

  corr_fold <- finestre_df[i, ]
  train_df <- dati_giocatore %>% filter(Date >= corr_fold$inizio_train & Date <= corr_fold$fine_train) %>% arrange(Date)
  test_df  <- dati_giocatore %>% filter(Date >= corr_fold$inizio_test  & Date <= corr_fold$fine_test)

  colonne_da_escludere <- c("point_won", "match_id", "unique_set_id", "unique_game_id", "Date",
                            "fatigue_focal_cum", "fatigue_opp_cum", "fatigue_focal_recent", "fatigue_opp_recent",
                            "prev_rally_length")
  y_train_full <- train_df$point_won
  y_test       <- test_df$point_won

  train_df_clean <- train_df
  test_df_clean <- test_df

  for(col in names(train_df_clean)) {
    if(is.numeric(train_df_clean[[col]])) {
      mediana <- median(train_df_clean[[col]], na.rm = TRUE)
      if (is.na(mediana)) mediana <- 0
      train_df_clean[[col]][is.na(train_df_clean[[col]])] <- mediana
```

```
30        test_df_clean[[col]][is.na(test_df_clean[[col]])] <- mediana
31      } else {
32        train_df_clean[[col]] <- as.character(train_df_clean[[col]])
33        train_df_clean[[col]][is.na(train_df_clean[[col]])] <- "Missing"
34        train_df_clean[[col]] <- as.factor(train_df_clean[[col]])
35
36        test_df_clean[[col]] <- as.character(test_df_clean[[col]])
37        test_df_clean[[col]][is.na(test_df_clean[[col]])] <- "Missing"
38      }
39    }
40
41    allinea_fattori_k <- function(df_train, df_target) {
42      for(col in names(df_train)) {
43        if(!is.numeric(df_train[[col]])) {
44          livelli_train <- levels(df_train[[col]])
45          idx_unseen <- !(df_target[[col]] %in% livelli_train)
46          if(any(idx_unseen)) {
47            fallback <- ifelse("Missing" %in% livelli_train, "Missing", livelli_
                train[1])
48            df_target[[col]][idx_unseen] <- fallback
49          }
50          df_target[[col]] <- factor(df_target[[col]], levels = livelli_train)
51        }
52      }
53      return(df_target)
54    }
55
56    test_df_clean <- allinea_fattori_k(train_df_clean, test_df_clean)
57
58    predittori_iniziali <- setdiff(names(train_df_clean), colonne_da_escludere)
59    predittori_validi <- c()
60    for(v in predittori_iniziali) {
61      if(length(unique(train_df_clean[[v]])) > 1) {
62        predittori_validi <- c(predittori_validi, v)
63      }
64    }
65
66    keras3::clear_session()
67
68    X_train_k <- model.matrix(~ . - 1, data = train_df_clean %>% select(all_of(
        predittori_validi)))
69    X_test_k  <- model.matrix(~ . - 1, data = test_df_clean %>% select(all_of(
        predittori_validi)))
70
71    missing_cols_k <- setdiff(colnames(X_train_k), colnames(X_test_k))
72    if(length(missing_cols_k) > 0) {
73      miss_mat_k <- matrix(0, nrow=nrow(X_test_k), ncol=length(missing_cols_k))
74      colnames(miss_mat_k) <- missing_cols_k
75      X_test_k <- cbind(X_test_k, miss_mat_k)
76    }
77    X_test_k  <- X_test_k[, colnames(X_train_k), drop = FALSE]
78
79    means_k <- colMeans(X_train_k, na.rm = TRUE)
80    sds_k   <- apply(X_train_k, 2, sd, na.rm = TRUE)
81    sds_k[sds_k == 0] <- 1
82
```

```
X_train_k_scaled <- scale(X_train_k, center = means_k, scale = sds_k)
X_test_k_scaled  <- scale(X_test_k, center = means_k, scale = sds_k)

match_ids <- unique(train_df_clean$match_id)
n_matches_train <- floor(length(match_ids) * 0.8)
set.seed(42 + i)
matches_train_tune <- sample(match_ids, size = n_matches_train, replace =
    FALSE)

idx_tune_train <- train_df_clean$match_id %in% matches_train_tune
idx_tune_val   <- !idx_tune_train

X_tune_train <- X_train_k_scaled[idx_tune_train, , drop=FALSE]
y_tune_train <- y_train_full[idx_tune_train]
X_tune_val   <- X_train_k_scaled[idx_tune_val, , drop=FALSE]
y_tune_val   <- y_train_full[idx_tune_val]

griglia_keras <- expand.grid(
  learning_rate = c(0.001, 0.0005),
  capacity      = c("Large", "Small"),
  dropout       = c("High", "Low"),
  stringsAsFactors = FALSE
)

best_auc_k <- -1
best_params_k <- NULL
best_epoch_k <- 100

for(k in 1:nrow(griglia_keras)) {
  p <- griglia_keras[k, ]
  keras3::clear_session()

  u1 <- ifelse(p$capacity == "Large", 128, 64)
  u2 <- ifelse(p$capacity == "Large", 64,  32)
  u3 <- ifelse(p$capacity == "Large", 32,  16)

  d1 <- ifelse(p$dropout == "High", 0.3, 0.2)
  d2 <- ifelse(p$dropout == "High", 0.2, 0.1)

  mod_tune <- keras_model_sequential(layers = list(
    layer_input(shape = c(ncol(X_train_k_scaled))),
    layer_dense(units = u1, activation = "relu"), layer_dropout(rate = d1),
    layer_dense(units = u2, activation = "relu"), layer_dropout(rate = d2),
    layer_dense(units = u3, activation = "relu"),
    layer_dense(units = 1,  activation = "sigmoid")
  ))

  mod_tune %>% compile(
    optimizer = optimizer_adam(learning_rate = p$learning_rate),
    loss = "binary_crossentropy", metrics = c("AUC")
  )

  early_stop_tune <- callback_early_stopping(monitor = "val_loss", patience =
      8, restore_best_weights = TRUE)

  storia <- mod_tune %>% fit(
```

```
      x = as.matrix(X_tune_train), y = as.numeric(y_tune_train),
      validation_data = list(as.matrix(X_tune_val), as.numeric(y_tune_val)),
      epochs = 100, batch_size = 256, callbacks = list(early_stop_tune), verbose = 0
    )

    pred_val_raw <- predict(mod_tune, as.matrix(X_tune_val), verbose = 0)
    auc_corrente <- as.numeric(pROC::roc(y_tune_val, as.numeric(pred_val_raw), quiet = TRUE)$auc)

    if(!is.na(auc_corrente) && auc_corrente > best_auc_k) {
      best_auc_k <- auc_corrente
      best_params_k <- p
      epoca_minima <- which.min(storia$metrics$val_loss)
      if(length(epoca_minima) > 0) best_epoch_k <- as.numeric(epoca_minima)
    }
  }

  keras3::clear_session()

  u1_fin <- ifelse(best_params_k$capacity == "Large", 128, 64)
  u2_fin <- ifelse(best_params_k$capacity == "Large", 64,  32)
  u3_fin <- ifelse(best_params_k$capacity == "Large", 32,  16)

  d1_fin <- ifelse(best_params_k$dropout == "High", 0.3, 0.2)
  d2_fin <- ifelse(best_params_k$dropout == "High", 0.2, 0.1)

  mod_final <- keras_model_sequential(layers = list(
    layer_input(shape = c(ncol(X_train_k_scaled))),
    layer_dense(units = u1_fin, activation = "relu"), layer_dropout(rate = d1_fin),
    layer_dense(units = u2_fin, activation = "relu"), layer_dropout(rate = d2_fin),
    layer_dense(units = u3_fin, activation = "relu"),
    layer_dense(units = 1,  activation = "sigmoid")
  ))

  mod_final %>% compile(
    optimizer = optimizer_adam(learning_rate = best_params_k$learning_rate),
    loss = "binary_crossentropy", metrics = c("AUC")
  )

  final_epochs_keras <- round(best_epoch_k * (10/8))
  if(final_epochs_keras < 1) final_epochs_keras <- 1

  mod_final %>% fit(
    x = as.matrix(X_train_k_scaled), y = as.numeric(y_train_full),
    epochs = final_epochs_keras, batch_size = 256, verbose = 0
  )

  pred_keras_raw <- predict(mod_final, as.matrix(X_test_k_scaled), verbose = 0)
  pred_keras <- as.numeric(pred_keras_raw)

  # Salviamo le previsioni per poterle passare al ciclo foreach
  keras_preds_list[[i]] <- pred_keras

```

```r
  auc_keras <- as.numeric(pROC::roc(y_test, pred_keras, quiet = TRUE)$auc)
  met_keras <- calc_metrics(y_test, pred_keras)

  risultati_keras <- rbind(risultati_keras, data.frame(
    Fold = i,
    Keras_LR = best_params_k$learning_rate,
    Keras_Capacity = best_params_k$capacity,
    Keras_Dropout = best_params_k$dropout,
    AUC_Keras = auc_keras,
    F1_Keras = met_keras["F1"],
    MCC_Keras = met_keras["MCC"]
  ))

  gc()
}

# ============================================================
# FASE 5: MODELLAZIONE WALK-FORWARD ALBERI E STATISTICA
# ============================================================
cat("\n--- FASE 5: Modellazione Walk-Forward (Parallelizzata, Singolo Thread Interno) ---\n")
flush.console()

num_cores <- max(1, detectCores()/2 - 1)
cl <- makeCluster(num_cores, outfile = "")
registerDoParallel(cl)

risultati_stat_trees <- foreach(
  i = 1:nrow(finestre_df),
  .combine = rbind,
  .export = c("keras_preds_list", "calc_metrics"), # Esporta la lista e la funzione!
  .packages = c("dplyr", "lubridate", "xgboost", "gpboost", "mgcv", "lme4", "glmertree", "partykit", "pROC", "Matrix", "stringr", "shapviz", "ggplot2")
) %dopar% {

  # === FUNZIONE DI LOGGING PER CONSOLE PARALLELA ===
  log_msg <- function(msg) {
    cat(sprintf("[%s] [FOLD %d] %s\n", format(Sys.time(), "%H:%M:%S"), i, msg))
  }

  log_msg("Inizio Preparazione Dati")

  corr_fold <- finestre_df[i, ]
  train_df <- dati_giocatore %>% filter(Date >= corr_fold$inizio_train & Date <= corr_fold$fine_train) %>% arrange(Date)
  test_df  <- dati_giocatore %>% filter(Date >= corr_fold$inizio_test  & Date <= corr_fold$fine_test)

  if(nrow(train_df) < 50 || nrow(test_df) < 10) {
    log_msg("ERRORE: Dati insufficienti in questa finestra temporale")
    return(data.frame(
      Fold = i,
      AUC_GLM_DC = NA, F1_GLM_DC = NA, MCC_GLM_DC = NA,
      AUC_GLM = NA, F1_GLM = NA, MCC_GLM = NA,
      AUC_GLMM = NA, F1_GLMM = NA, MCC_GLMM = NA,
```

```
      AUC_GAMM = NA, F1_GAMM = NA, MCC_GAMM = NA,
      AUC_GLM_Tree = NA, F1_GLM_Tree = NA, MCC_GLM_Tree = NA,
      AUC_GLMM_Tree = NA, F1_GLMM_Tree = NA, MCC_GLMM_Tree = NA,
      AUC_XGBoost = NA, F1_XGBoost = NA, MCC_XGBoost = NA,
      AUC_GPBoost = NA, F1_GPBoost = NA, MCC_GPBoost = NA
    ))
  }

  colonne_da_escludere <- c("point_won", "match_id", "unique_set_id", "unique_game_id", "Date",
                            "fatigue_focal_cum", "fatigue_opp_cum", "fatigue_focal_recent", "fatigue_opp_recent",
                            "prev_rally_length")

  # Matrici grezze
  X_train_full <- model.matrix(~ . - 1, data = train_df %>% select(-any_of(colonne_da_escludere)), na.action = "na.pass")
  X_test_mat   <- model.matrix(~ . - 1, data = test_df %>% select(-any_of(colonne_da_escludere)), na.action = "na.pass")
  y_train_full <- train_df$point_won
  y_test       <- test_df$point_won

  missing_cols <- setdiff(colnames(X_train_full), colnames(X_test_mat))
  if(length(missing_cols) > 0) {
    miss_mat <- matrix(0, nrow = nrow(X_test_mat), ncol = length(missing_cols))
    colnames(miss_mat) <- missing_cols
    X_test_mat <- cbind(X_test_mat, miss_mat)
  }
  X_test_mat <- X_test_mat[, colnames(X_train_full), drop = FALSE]

  match_ids <- unique(train_df$match_id)
  n_matches_train <- floor(length(match_ids) * 0.7)
  set.seed(42 + i)
  matches_train_tune <- sample(match_ids, size = n_matches_train, replace = FALSE)
  idx_train_tune <- train_df$match_id %in% matches_train_tune

  # NTHREAD = 1 PER EVITARE SOVRACCARICO
  dtrain_tune <- xgb.DMatrix(data = X_train_full[idx_train_tune, , drop=FALSE], label = y_train_full[idx_train_tune], nthread = 1)
  dval_tune   <- xgb.DMatrix(data = X_train_full[!idx_train_tune, , drop=FALSE], label = y_train_full[!idx_train_tune], nthread = 1)

  # ------------------------------------------------------------
  # PREPARAZIONE DATI
  # ------------------------------------------------------------
  train_df_clean <- train_df
  test_df_clean <- test_df

  for(col in names(train_df_clean)) {
    if(is.numeric(train_df_clean[[col]])) {
      mediana <- median(train_df_clean[[col]], na.rm = TRUE)
      if (is.na(mediana)) mediana <- 0
      train_df_clean[[col]][is.na(train_df_clean[[col]])] <- mediana
      test_df_clean[[col]][is.na(test_df_clean[[col]])] <- mediana
    } else {
```

```r
      train_df_clean[[col]] <- as.character(train_df_clean[[col]])
      train_df_clean[[col]][is.na(train_df_clean[[col]])] <- "Missing"
      train_df_clean[[col]] <- as.factor(train_df_clean[[col]])
      test_df_clean[[col]] <- as.character(test_df_clean[[col]])
      test_df_clean[[col]][is.na(test_df_clean[[col]])] <- "Missing"
    }
  }

  allinea_fattori <- function(df_train, df_target) {
    for(col in names(df_train)) {
      if(!is.numeric(df_train[[col]])) {
        livelli_train <- levels(df_train[[col]])
        idx_unseen <- !(df_target[[col]] %in% livelli_train)
        if(any(idx_unseen)) {
          fallback <- ifelse("Missing" %in% livelli_train, "Missing", livelli_
              train[1])
          df_target[[col]][idx_unseen] <- fallback
        }
        df_target[[col]] <- factor(df_target[[col]], levels = livelli_train)
      }
    }
    return(df_target)
  }

  test_df_clean <- allinea_fattori(train_df_clean, test_df_clean)

  predittori_iniziali <- setdiff(names(train_df_clean), colonne_da_escludere)
  predittori_validi <- c()
  for(v in predittori_iniziali) {
    if(length(unique(train_df_clean[[v]])) > 1) predittori_validi <- c(
        predittori_validi, v)
  }

  train_df_clean$match_id <- as.factor(train_df_clean$match_id)
  train_df_clean$unique_set_id <- as.factor(train_df_clean$unique_set_id)
  test_df_clean$match_id <- factor(test_df_clean$match_id, levels = levels(train
      _df_clean$match_id))
  test_df_clean$unique_set_id <- factor(test_df_clean$unique_set_id, levels =
      levels(train_df_clean$unique_set_id))

  train_df_tune <- train_df_clean[idx_train_tune, ]
  val_df_tune   <- train_df_clean[!idx_train_tune, ]
  for(col in names(train_df_tune)) { if(is.factor(train_df_tune[[col]])) train_
      df_tune[[col]] <- droplevels(train_df_tune[[col]]) }
  val_df_tune <- allinea_fattori(train_df_tune, val_df_tune)

  # ============================================================
  # 1: BASELINE: GLM E GLMM
  # ============================================================
  log_msg("Addestramento GLM / GLMM")
  formula_glm <- as.formula(paste("point_won ~", paste(predittori_validi,
      collapse = " + ")))

  suppressWarnings({
    mod_glm <- glm(formula_glm, family = binomial(link = "logit"), data = train_
        df_clean)
```

```
338     mod_glm_dc <- glm(point_won ~ Player_Baseline_Prob, family = binomial(link =
            "logit"), data = train_df_clean)
339   })
340
341   formula_glmm <- as.formula(paste("point_won ~", paste(predittori_validi,
          collapse = " + "), "+ (1 | match_id)"))
342   mod_glmm <- tryCatch({
343     glmer(formula_glmm, family = binomial(link = "logit"), data = train_df_clean
            ,
344           control = glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2
                e5)))
345   }, error = function(e) { NULL })
346
347   # ============================================================
348   # 2: TUNING E FIT GLM TREE
349   # ============================================================
350   log_msg("Addestramento GLM Tree")
351   mob_formula_str <- paste("point_won ~ 1 |", paste(predittori_validi, collapse
          = " + "))
352   mob_formula <- as.formula(mob_formula_str)
353
354   griglia_mob <- expand.grid(alpha = c(0.05, 0.01), maxdepth = c(3, 4))
355   best_auc_mob <- -1
356   best_params_mob <- list(alpha = 0.05, maxdepth = 3)
357
358   for(k in 1:nrow(griglia_mob)) {
359     p <- griglia_mob[k, ]
360     mod_tune_mob <- tryCatch({
361       partykit::glmtree(mob_formula, data = train_df_tune, family = binomial(
              link = "logit"),
362                         alpha = p$alpha, maxdepth = p$maxdepth)
363     }, error = function(e) { NULL })
364     if(!is.null(mod_tune_mob)) {
365       pred_val_mob <- predict(mod_tune_mob, newdata = val_df_tune, type = "
              response")
366       auc_corrente <- as.numeric(pROC::roc(val_df_tune$point_won, pred_val_mob,
              quiet = TRUE)$auc)
367       if(!is.na(auc_corrente) && auc_corrente > best_auc_mob) {
368         best_auc_mob <- auc_corrente
369         best_params_mob <- list(alpha = p$alpha, maxdepth = p$maxdepth)
370       }
371     }
372   }
373   mod_mob_final <- tryCatch({
374     partykit::glmtree(mob_formula, data = train_df_clean, family = binomial(link
            = "logit"),
375                       alpha = best_params_mob$alpha, maxdepth = best_params_mob$
                          maxdepth)
376   }, error = function(e) { NULL })
377
378   # ============================================================
379   # 3: TUNING E FIT GLMM TREES
380   # ============================================================
381   log_msg("Addestramento GLMM Tree")
382   tree_formula_str <- paste("point_won ~ 1 | (1 | match_id) |", paste(predittori
          _validi, collapse = " + "))
```

```
tree_formula <- as.formula(tree_formula_str)

griglia_glmmtree <- expand.grid(alpha = c(0.05, 0.01), maxdepth = c(3, 4))
best_auc_glmmtree <- -1
best_params_glmmtree <- list(alpha = 0.05, maxdepth = 3)

for(k in 1:nrow(griglia_glmmtree)) {
  p <- griglia_glmmtree[k, ]
  mod_tune_glmmtree <- tryCatch({
    glmertree(tree_formula, data = train_df_tune, family = binomial(link = "
        logit"),
              dfsplit = TRUE, alpha = p$alpha, maxdepth = p$maxdepth)
  }, error = function(e) { NULL })
  if(!is.null(mod_tune_glmmtree)) {
    pred_val_glmmtree <- predict(mod_tune_glmmtree, newdata = val_df_tune,
        type = "response", re.form = NA, allow.new.levels = TRUE)
    auc_corrente <- as.numeric(pROC::roc(val_df_tune$point_won, pred_val_
        glmmtree, quiet = TRUE)$auc)
    if(!is.na(auc_corrente) && auc_corrente > best_auc_glmmtree) {
      best_auc_glmmtree <- auc_corrente
      best_params_glmmtree <- list(alpha = p$alpha, maxdepth = p$maxdepth)
    }
  }
}
mod_glmmtree_final <- tryCatch({
  glmertree(tree_formula, data = train_df_clean, family = binomial(link = "
      logit"),
            alpha = best_params_glmmtree$alpha, maxdepth = best_params_
                glmmtree$maxdepth)
}, error = function(e) { NULL })

# ============================================================
# 4: TUNING XGBOOST
# ============================================================
log_msg("Addestramento XGBoost")
griglia_xgb <- expand.grid(
  max_depth = c(2, 4, 6),
  eta = c(0.01, 0.05),
  min_child_weight = c(15, 30, 50),
  colsample_bytree = c(0.75, 1.0)
)
best_auc_xgb <- -1
best_params_xgb <- list()
watchlist_xgb <- list(train = dtrain_tune, validation = dval_tune)

for(k in 1:nrow(griglia_xgb)) {
  p <- griglia_xgb[k, ]
  param_xgb <- xgb.params(
    objective = "binary:logistic",
    eval_metric = "auc",
    max_depth = p$max_depth,
    eta = p$eta,
    min_child_weight = p$min_child_weight,
    colsample_bytree = p$colsample_bytree,
    alpha = 1, lambda = 5,
    nthread = 1
```

```
    )
    mod_tune_xgb <- tryCatch({
      xgb.train(param_xgb, data = dtrain_tune, nrounds = 1500, early_stopping_
          rounds = 40, maximize = TRUE, evals = watchlist_xgb, verbose = 0)
    }, error=function(e) NULL)

    if (!is.null(mod_tune_xgb) && !is.null(xgb.attr(mod_tune_xgb,"best_score"))
        && (xgb.attr(mod_tune_xgb,"best_score") > best_auc_xgb)) {
      best_auc_xgb <- xgb.attr(mod_tune_xgb,"best_score")
      best_params_xgb <- list(max_depth = p$max_depth, eta = p$eta, min_child_
          weight = p$min_child_weight, colsample_bytree = p$colsample_bytree,
          nrounds = xgb.attr(mod_tune_xgb,"best_iteration"))
    }
  }

  # ============================================================
  # 5: TUNING GPBOOST
  # ============================================================
  log_msg("Addestramento GPBoost")
  dtrain_tune_gpb <- gpb.Dataset(data = X_train_full[idx_train_tune, , drop=
      FALSE], label = y_train_full[idx_train_tune], free_raw_data = FALSE)
  dval_tune_gpb   <- gpb.Dataset(data = X_train_full[!idx_train_tune, , drop=
      FALSE], label = y_train_full[!idx_train_tune], free_raw_data = FALSE)
  group_data_full <- train_df %>% select(match_id, unique_set_id, unique_game_id
      )
  group_train_tune <- group_data_full[idx_train_tune, ]
  group_val_tune   <- group_data_full[!idx_train_tune, ]

  griglia_gpb <- expand.grid(
    max_depth = c(2,4,6),
    learning_rate = c(0.01, 0.05),
    min_data_in_leaf = c(15, 30, 50),
    feature_fraction = c(0.75,1),
    lambda_l1 = c(0, 0.1),
    lambda_l2 = c(0.1, 1)
  )
  best_auc_gpb <- -1
  best_params_gpb <- list()

  for(k in 1:nrow(griglia_gpb)) {
    p <- griglia_gpb[k, ]
    gp_model_tune <- GPModel(group_data = group_train_tune, likelihood = "
        bernoulli_logit")
    gp_model_tune$set_prediction_data(group_data_pred = group_val_tune)
    params_tune_gpb <- list(
      learning_rate = p$learning_rate,
      max_depth = p$max_depth,
      min_data_in_leaf = p$min_data_in_leaf,
      feature_fraction = p$feature_fraction,
      lambda_l1 = p$lambda_l1,
      lambda_l2 = p$lambda_l2,
      metric = "auc",
      num_threads = 1
    )

    mod_tune_gpb <- tryCatch({
```

```
      gpb.train(params = params_tune_gpb, data = dtrain_tune_gpb, nrounds =
          1500,
            gp_model = gp_model_tune, use_gp_model_for_validation = TRUE,
            valids = list(validation = dval_tune_gpb), early_stopping_rounds
                = 40,
            train_gp_model_cov_pars = FALSE, verbose = 0)
  }, error=function(e) NULL)

  if (!is.null(mod_tune_gpb) && !is.null(mod_tune_gpb$best_score) && mod_tune_
      gpb$best_score > best_auc_gpb) {
    best_auc_gpb <- mod_tune_gpb$best_score

    best_params_gpb <- list(max_depth = p$max_depth, eta = p$learning_rate,
                            min_data_in_leaf = p$min_data_in_leaf,
                            feature_fraction = p$feature_fraction,
                            lambda_l1 = p$lambda_l1, lambda_l2 = p$lambda_l2,
                            nrounds = mod_tune_gpb$best_iter)
  }
}

# ============================================================
# 6: PREPARAZIONE E FIT GAMM
# ============================================================
log_msg("Addestramento GAMM")
form_terms <- c()
for(v in predittori_validi) {
  if(is.numeric(train_df_clean[[v]]) && length(unique(train_df_clean[[v]])) >=
      10) {
    form_terms <- c(form_terms, paste0("s(", v, ", bs='tp', k=5)"))
  } else {
    form_terms <- c(form_terms, v)
  }
}
form_str <- paste("point_won ~", paste(form_terms, collapse = " + "))
form_str <- paste(form_str, "+ s(match_id, bs='re') + s(unique_set_id, bs='re
    ')")
formula_gamm <- as.formula(form_str)
mod_gamm <- bam(formula_gamm, data = train_df_clean, family = binomial(link =
    "logit"), method = "fREML", nthreads = 1)

# ============================================================
# STIMA FINALE E PREVISIONI E METRICHE
# ============================================================
log_msg("Calcolo delle Previsioni e Metriche sul Test Set")
suppressWarnings({
  pred_glm_raw <- predict(mod_glm, newdata = test_df_clean, type = "response")
  pred_glm_dc_raw <- predict(mod_glm_dc, newdata = test_df_clean, type = "
      response")
})
pred_glm <- as.numeric(pred_glm_raw)
pred_glm_dc <- as.numeric(pred_glm_dc_raw)

if(!is.null(mod_glmm)) {
  pred_glmm <- as.numeric(predict(mod_glmm, newdata = test_df_clean, type = "
      response", re.form = NA, allow.new.levels = TRUE))
} else {
```

```
    pred_glmm <- rep(NA, nrow(test_df_clean))
  }

  if(!is.null(mod_mob_final)) {
    pred_mob <- as.numeric(predict(mod_mob_final, newdata = test_df_clean, type
        = "response"))
  } else {
    pred_mob <- rep(NA, nrow(test_df_clean))
  }

  if(!is.null(mod_glmmtree_final)) {
    pred_glmmtree <- as.numeric(predict(mod_glmmtree_final, newdata = test_df_
        clean, type = "response", re.form = NA, allow.new.levels = TRUE))
  } else {
    pred_glmmtree <- rep(NA, nrow(test_df_clean))
  }

  # --- XGBoost Finale ---
  final_rounds_xgb <- round(best_params_xgb$nrounds)
  if(final_rounds_xgb < 10) final_rounds_xgb <- 10
  dtrain_full_xgb <- xgb.DMatrix(data = X_train_full, label = y_train_full,
      nthread = 1)
  param_xgb_final <- xgb.params(
    objective = "binary:logistic",
    eval_metric = "auc",
    max_depth = best_params_xgb$max_depth,
    eta = best_params_xgb$eta,
    min_child_weight = best_params_xgb$min_child_weight,
    colsample_bytree = best_params_xgb$colsample_bytree,
    alpha = 1, lambda = 5,
    nthread = 1
  )
  mod_xgb <- xgb.train(param_xgb_final, data = dtrain_full_xgb, nrounds = final_
      rounds_xgb, verbose = 0)
  pred_xgb <- predict(mod_xgb, X_test_mat)

  # --- GPBoost Finale ---
  final_rounds_gpb <- round(best_params_gpb$nrounds)
  if(final_rounds_gpb < 10) final_rounds_gpb <- 10
  dtrain_full_gpb <- gpb.Dataset(data = X_train_full, label = y_train_full)
  gp_model_final <- GPModel(group_data = group_data_full, likelihood = "
      bernoulli_logit")
  mod_gpb_final <- gpboost(
    data = dtrain_full_gpb,
    gp_model = gp_model_final,
    nrounds = final_rounds_gpb,
    params = list(
      learning_rate = best_params_gpb$eta,
      max_depth = best_params_gpb$max_depth,
      min_data_in_leaf = best_params_gpb$min_data_in_leaf,
      feature_fraction = best_params_gpb$feature_fraction,
      lambda_l1 = best_params_gpb$lambda_l1,
      lambda_l2 = best_params_gpb$lambda_l2,
      num_threads = 1
    ),
    verbose = 0
```

```
  )
  pred_gpb_obj <- predict(mod_gpb_final, data = X_test_mat, group_data_pred =
      test_df %>% select(match_id, unique_set_id, unique_game_id), predict_var =
       FALSE, pred_latent = FALSE)
  pred_gpb <- pred_gpb_obj$response_mean

  # --- GAMM Finale ---
  pred_gamm_raw <- predict(mod_gamm, newdata = test_df_clean, type = "response",
       exclude = c("s(match_id)", "s(unique_set_id)"))
  pred_gamm <- as.numeric(pred_gamm_raw)

  # --- Recupero previsioni KERAS ---
  pred_keras <- keras_preds_list[[i]]

  # --- Calcolo Metriche (AUC, F1, MCC) ---
  auc_glm_dc <- as.numeric(pROC::roc(y_test, pred_glm_dc, quiet = TRUE)$auc)
  met_glm_dc <- calc_metrics(y_test, pred_glm_dc)

  auc_glm <- as.numeric(pROC::roc(y_test, pred_glm, quiet = TRUE)$auc)
  met_glm <- calc_metrics(y_test, pred_glm)

  auc_glmm <- if(all(is.na(pred_glmm))) NA else as.numeric(pROC::roc(y_test,
      pred_glmm, quiet = TRUE)$auc)
  met_glmm <- calc_metrics(y_test, pred_glmm)

  auc_mob <- if(all(is.na(pred_mob))) NA else as.numeric(pROC::roc(y_test, pred_
      mob, quiet = TRUE)$auc)
  met_mob <- calc_metrics(y_test, pred_mob)

  auc_glmmtree <- if(all(is.na(pred_glmmtree))) NA else as.numeric(pROC::roc(y_
      test, pred_glmmtree, quiet = TRUE)$auc)
  met_glmmtree <- calc_metrics(y_test, pred_glmmtree)

  auc_xgb <- as.numeric(pROC::roc(y_test, pred_xgb, quiet = TRUE)$auc)
  met_xgb <- calc_metrics(y_test, pred_xgb)

  auc_gpb <- as.numeric(pROC::roc(y_test, pred_gpb, quiet = TRUE)$auc)
  met_gpb <- calc_metrics(y_test, pred_gpb)

  auc_gamm <- as.numeric(pROC::roc(y_test, pred_gamm, quiet = TRUE)$auc)
  met_gamm <- calc_metrics(y_test, pred_gamm)

  log_msg("Fold Terminato con successo!\n")

  # Salvataggio di AUC, F1 e MCC per ogni modello
  data.frame(
    Fold = i,
    AUC_GLM_DC = auc_glm_dc, F1_GLM_DC = met_glm_dc["F1"], MCC_GLM_DC = met_glm_
        dc["MCC"],
    AUC_GLM = auc_glm,       F1_GLM = met_glm["F1"],       MCC_GLM = met_glm["
        MCC"],
    AUC_GLMM = auc_glmm,     F1_GLMM = met_glmm["F1"],     MCC_GLMM = met_glmm["
        MCC"],
    AUC_GAMM = auc_gamm,     F1_GAMM = met_gamm["F1"],     MCC_GAMM = met_gamm["
        MCC"],
    AUC_GLM_Tree = auc_mob,  F1_GLM_Tree = met_mob["F1"],  MCC_GLM_Tree = met_
```

```
          mob["MCC"],
627     AUC_GLMM_Tree = auc_glmmtree, F1_GLMM_Tree = met_glmmtree["F1"], MCC_GLMM_
          Tree = met_glmmtree["MCC"],
628     AUC_XGBoost = auc_xgb,   F1_XGBoost = met_xgb["F1"],   MCC_XGBoost = met_xgb
          ["MCC"],
629     AUC_GPBoost = auc_gpb,   F1_GPBoost = met_gpb["F1"],   MCC_GPBoost = met_gpb
          ["MCC"],
630     row.names = NULL
631   )
632 }
633 stopCluster(cl)
634
635 diff <- difftime(Sys.time(), inizio_totale, units = "secs")
636 cat(paste("\nTempo totale Walk-Forward:", floor(as.numeric(diff)/60), "minuti e"
      , round(as.numeric(diff)%%60), "secondi.\n"))
637
638 # Merge dei risultati con quelli di Keras
639 risultati_finali <- risultati_stat_trees %>% left_join(risultati_keras, by = "
      Fold")
```

Elenco delle figure

Elenco delle tabelle

Bibliografia

Azzalini, A., e Scarpa, B. (2012). *Data analysis and data mining: An introduction*. Oxford: Oxford University Press [22, 24].

Bates, D., Mächler, M., Bolker, B., e Walker, S. (2015). "Fitting Linear Mixed-Effects Models Using lme4". In: *Journal of Statistical Software* 67.1, pp. 1–48 [25, 30].

Breslow, N. E., e Clayton, D. G. (1993). "Approximate inference in generalized linear mixed models". In: *Journal of the American Statistical Association* 88.421, pp. 9–25 [25–27].

Chen, T., e Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794 [4, 32].

Chicco, D., e Jurman, G. (2020). "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC genomics* 21.1, pp. 1–13 [37].

Chollet, F. et al. (2015). *Keras*. URL: `https://keras.io` (visitato il giorno 20/01/2026) [34].

Dixon, M. J., e Coles, S. G. (1997). "Modelling association football scores and inefficiencies in the football betting market". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 46.2, pp. 265–280 [13, 24, 38, 47].

Fokkema, M., Edbrooke-Childs, J., e Wolpert, M. (2021). "Generalized linear mixed-model (GLMM) trees: A flexible decision-tree method for multilevel and longitudinal data". In: *Psychotherapy Research* 31.3, pp. 329–341 [29, 30].

Freund, Y., e Schapire, R. E. (1997). "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of computer and system sciences* 55.1, pp. 119–139 [31].

Friedman, J. H. (2001). "Greedy function approximation: a gradient boosting machine". In: *Annals of Statistics*, pp. 1189–1232 [32].

Hajjem, A., Bellavance, F., e Larocque, D. (2014). "Mixed-effects random forest for clustered data". In: *Journal of Statistical Computation and Simulation* 84.6, pp. 1313–1328 [31].

Hastie, T., e Tibshirani, R. (1990). "Exploring the Nature of Covariate Effects in the Proportional Hazards Model". In: *Biometrics* 46.4, pp. 1005–1016 [26].

Hoerl, A. E., e Kennard, R. W. (1970). "Ridge regression: Biased estimation for nonorthogonal problems". In: *Technometrics* 12.1, pp. 55–67 [13].

Hovad, E., Mikkelsen, H. C., e Illum, M. (2025). *Analysis of points outcome in ATP Grand Slam Tennis using big data and machine learning*. arXiv: 2506.05866 [4].

Kingma, D. P., e Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv: 1412.6980 [36].

Kovalchik, S. A., e Reid, M. (2016). "Measuring clutch performance in professional tennis". In: *Statistica Applicata-Italian Journal of Applied Statistics* 30.2 [5, 12].

Lin, X., e Zhang, D. (1999). "Inference in generalized additive mixed models by using smoothing splines". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.2, pp. 381–400 [26].

Lundberg, S. M., e Lee, S.-I. (2017). "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 4765–4774 [45].

Makino, M., Odaka, T., Kuroiwa, J., Suwa, I., e Shirai, H. (2020). "Feature Selection to Win the Point of ATP Tennis Players Using Rally Information". In: *International Journal of Computer Science in Sport* 19.1, pp. 43–58 [5].

Nelder, J. A., e Wedderburn, R. W. (1972). "Generalized linear models". In: *Journal of the Royal Statistical Society: Series A (General)* 135.3, pp. 370–384 [25].

Pedersen, E. J., Miller, D. L., Simpson, G. L., e Ross, N. (2019). "Hierarchical generalized additive models in ecology: an introduction with mgcv". In: *PeerJ* 7, e6876 [27].

Ruppert, D., Wand, M. P., e Carroll, R. J. (2003). *Semiparametric Regression*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press [28].

Sackmann, J. (2025). *The Tennis Abstract Match Charting Project*. URL: https://github.com/JeffSackmann/tennis_MatchChartingProject (visitato il giorno 02/11/2025) [7].

Sigrist, F. (2022). "Gaussian Process Boosting". In: *Journal of Machine Learning Research* 23.232, pp. 1–46 [6, 31, 33, 51].

Sipko, M., e Knottenbelt, W. (2015). "Machine learning for the prediction of professional tennis matches". MEng Computing Final Year Project. Imperial College London [4].

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., e Salakhutdinov, R. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1, pp. 1929–1958 [35].

Vives, F., Lázaro, J., Guzmán, J. F., Crespo, M., e Martínez-Gallego, R. (2024). "Machine Learning in Tennis". In: *Racket Sports Analytics*. Springer, pp. 191–205 [4].

Wei, X., Lucey, P., Morgan, S., Reid, M., e Sridharan, S. (2016). "The thin edge of the wedge: Accurately predicting shot outcomes in tennis using style and context priors". In: *MIT Sloan Sports Analytics Conference* [5].

Wood, S. N. (2011). “Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.1, pp. 3–36 [27].

Wood, S. N. (2017). *Generalized Additive Models: An Introduction with R*. 2ª ed. Boca Raton, FL: Chapman e Hall/CRC [26, 28].

Wood, S. N., Goude, Y., e Shaw, S. (2015). “Generalized additive models for large data sets”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 64.1, pp. 139–155 [27].

Youden, W. J. (1950). “Index for rating diagnostic tests”. In: *Cancer* 3.1, pp. 32–35 [38].