POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Informatica
Data Analytics

Tesi di Laurea Magistrale

Design, studio, modelizzazione di nuove metriche e KPI per analisi giocatori di Tennis.

Relatore
prof. Paolo Garza
Supervisore Aziendale
Deltatre s.p.a
dott. Federico Barbiero

Candidato
Matteo Gally

Anno Accademico 2021-2022

Sommario

L'obiettivo di questo progetto è quello di creare una piattaforma che permetta in modo semplice e intuitivo di ottenere in tempo reale delle metriche e degli indicatori che consentano di misurare le performance di un giocatore o poterne mettere a confronto due. Lo sviluppo del progetto si articola in tre macrofasi, la prima consiste nello studio e nella creazione di un data-warehouse che possa immagazzinare tutti i dati che servono, la seconda fase invece riguarda l'estrazione dei file json forniti come dataset, dove i dati vengono elaborati fino ad essere caricati nel database, mentre l'ultima fase concerne l'aggregazione dei dati fino ad ottenere metriche, indici e grafiche di interesse.

Infine, è stata realizzata un'applicazione web che incapsula tutto l'applicativo e permette di analizzare in tempo reale gli indici calcolati sui dati delle partite, restituendo agli utenti grafiche riassuntive di quanto calcolato in precedenza.
L'applicativo permette di fornire dei parametri passati dall'utente mediante l'utilizzo di menù a tendina e filtri di ricerca, così da ottenere una ricerca e/o uno studio più puntuale.

Ringraziamenti

Ringrazio il professor Paolo Garza per la disponibilità concessa per lo sviluppo di questo progetto.

Volevo inoltre ringraziare l'azienda Deltatre, specialmente Federico e tutto l'innovation lab, per avermi concesso questa opportunità, per avermi seguito ed essere stati disponibili fin da subito.

Ringrazio i miei amici, con cui ho condiviso momenti felici e che mi hanno aiutato ad alleggerire i periodi di esame e di lezione, strappandomi un sorriso.

Un grazie doveroso ai miei compagni di avventura Lukas, Carlo e Alessio, con cui ho condiviso gioie e dolori del percorso universitario, che mi hanno aiutato quando ero in difficoltà e con cui ho collaborato in diversi progetti, alla fine ne siamo usciti tutti quanti integri.

Ma soprattutto ringrazio la mia famiglia, i miei genitori in particolare, i quali fin dal principio sono stati fonte di ispirazione, supportandomi sempre e senza mai chiedere più di quel che stavo facendo, spero quindi che in questo traguardo oltre la felicità per me, trovino anche una soddisfazione personale.∞

Infine il più importante, grazie *Giulia* per tutto il tempo che mi hai dedicato.
Grazie, perché questo traguardo è anche merito tuo, grazie per il supporto nei momenti complicati e alla tua pazienza quando ascoltavi tutti i miei problemi o le idee che mi frullavano in testa.
Grazie perché posso dire di aver finalmente raggiunto questo primo grande traguardo, e spero che tu possa accompagnarmi ancora per molti altri.♡

Indice

Elenco delle tabelle v

Elenco delle figure vi

1 Introduzione 1

2 Librerie e Strumenti 2
- 2.1 Python . . . 2
- 2.2 MariaDB . . . 4
- 2.3 SQL . . . 5
- 2.4 MongoDB . . . 6

3 Stato attuale e Obbiettivi 7
- 3.1 Evoluzione dell'analisi . . . 7
- 3.2 Cos'è la Performance Analysis . . . 9
- 3.3 Data Analyst e Data Scientist nello sport . . . 9
- 3.4 Tennis e dati . . . 10
- 3.5 Deltatre . . . 12
  - 3.5.1 Chi è? . . . 12
  - 3.5.2 Come raccoglie i dati oggi . . . 13
- 3.6 Come vengono utilizzati i dati oggi . . . 13

4 Tipologia e struttura dati 16
- 4.1 Tipologie di dati . . . 16
- 4.2 Tipologia di variabili . . . 17
  - 4.2.1 Fondamentali . . . 17
  - 4.2.2 Altre tipologie di colpi . . . 19
- 4.3 Contenuto Dataset . . . 20

5 Progettazione del datawarehouse e sviluppo dell'applicazione 24
5.1 Definizione del problema 24
5.2 Pre-Processamento dei dati 25
5.3 Estrazione dei dati, da file Json a CSV 26
5.4 Creazione del Datawarehouse 29
5.5 Creazione dei Dati Raw attraverso le viste SQL 33
5.6 Creazione del dato aggregato e dei nuovi Indici/KPI 39
5.6.1 Serve Index 42
5.6.2 Return Index 43
5.6.3 Volley Index 44
5.6.4 Break Index 45
5.6.5 Overall Index 46
5.7 Creazione delle grafiche comparative 50
5.7.1 Serve chart 51
5.7.2 Return chart 52
5.7.3 Break chart 54
5.7.4 Volley chart 55
5.7.5 Overall chart 56
5.8 Creazione interfaccia grafica e applicazione web 57

6 Risultati e Casi d'uso 60

7 Conclusioni e Lavori Futuri 63

Elenco delle tabelle

5.1 Tabella esempio creazione indice 41
5.2 Tabella Serve Index 42
5.3 Tabella 2 Serve Index 42
5.4 Tabella 3 Serve Index 42
5.5 Tabella Return Index 43
5.6 Tabella 2 Return Index 43
5.7 Tabella 3 Return Index 43
5.8 Tabella Volley Index 44
5.9 Tabella 2 Volley Index 44
5.10 Tabella Break Index 45
5.11 Tabella 2 Break Index 45
5.12 Tabella Overall Index 46
5.13 Tabella 2 Overall Index 46
5.14 Tabella 3 Overall Index 46

Elenco delle figure

3.1 esempio di infografica di un match di tennis 14
3.2 esempio di infografica di riepilogo forma di un giocatore . . . 14
5.1 figura che mostra le differenze tra blocchi, a destra abbiamo un pezzo del file json riguardante il punto "corrente" mentre a sinistra lo stesso pezzo, ma riguardante il punto "precedente" 27
5.2 esempio di file .txt prodotto dall'estrazione 28
5.3 struttura e grafico del datawarehouse 33
5.4 grafica relativa al Serve index 52
5.5 grafica relativa al Return index 53
5.6 grafica relativa al Break index 54
5.7 grafica relativa al Volley index 55
5.8 grafica relativa all'Overall index 56
5.9 Screenshot della schermata principale dell'applicazione . . . 57

Capitolo 1

Introduzione

Al giorno di oggi le partite di tennis hanno una rappresentazione grafica molto scarna, facendo si che i pochi dati che ci vengono proposti a schermo risultino poco chiari o indecifrabili all'occhio di uno spettatore poco avvezzo al gioco del tennis.

Per questo motivo la società Deltatre con cui ho collaborato in questi mesi si sta impegnando per riuscire a trovare un modo per far si che vi sia la possibilità di raccontare l'andamento di una partita in maniera più 'interessante' e che possa descrivere più facilmente l'andamento di un match attraverso avvisi, grafiche o effetti speciali.

Al giorno d'oggi tutte le informazioni che abbiamo, ad esempio sullo stato di forma di un giocatore, le statistiche che ha nel confronto diretto con l'avversario, etc. passano tutte dal commentatore TV, infatti è lui che prima dell'incontro si informa sulle statistiche più interessanti e recenti che riguardano i due giocatori e sarà compito suo informare il pubblico che sta guardando l'incontro o sentendo per radio.

Deltatre quindi vuole quindi innovare andando a creare una un sistema che aiuti i sistemi televisivi e non, andando a fornire delle statistiche in tempo reale ai provider che possano aiutare il pubblico a seguire in maniera più efficace il match, ma che allo stesso tempo possano fornire uno strumento di analisi delle partite o di un giocatore professionale.

Capitolo 2

Librerie e Strumenti

2.1 Python

Come linguaggio di programmazione ho utilizzato pyhton, un linguaggio di 'alto livello' orientato a oggetti, adatto, tra gli altri usi, a sviluppare applicazioni per computazione numerica e scripting. [2]

Le origini del linguaggio Python risalgono al 1989, dove un ricercatore che lavorava nel National Research Institute for Mathematics and Computer Science di Amsterdam, Guido Van Rossum, ne inizia lo sviluppo.
Il nome fu evidentemente ispirato da una serie televisiva che la BBC trasmetteva agli inizi degli anni 70 "The Monty Python's Flying Circus".

L'idea era quella di implementare un linguaggio di programmazione che potesse essere facilmente appreso e utilizzato da persone che non avessero alcuna esperienza o conoscenza di progettazione e sviluppo software.[1]

Negli anni python si sta ritagliando sempre più spazio, sopratutto nell'ambito del data analytics, data science, machine learning e artificial intelligence. Infatti grazie alla sua natura open-source e ad alcune librerie di terze parti, come TensorFlow o Keras, si è in grado di costruire reti neurali o implementare modelli classificazione o regressione.

Per questo la scelta di questo linguaggio si è resa necessaria, grazie al fatto che il linguaggio, così modulabile, si è prestato molto bene all'integrazione di altre librerie di elevata utilità, e dall'alto potenziale come:

- **Pandas**: libreria che definisce il concetto di DataFrame, una sorta di struttura tabellare, indicizzata, che permette di essere manipolata in maniera simile ad un database SQL, ed inoltre permette di leggere e memorizzare il contenuto di file .csv e Excel, motivo per cui ho utilizzato soventemente questa libreria, impiegandola come variabile in cui ho salvato il risultato delle query SQL applicando alcune operazioni quali groupby, sum e max relative alle feature del database.

- **Streamlit**: è una libreria open-source di python che permette di creare facilmente delle applicazioni web personalizzate per il machine learning e la data science.[3]
  Nel mio progetto ho utilizzato la libreria per realizzare tutto il front-end, ovvero l'interfaccia web che permette all'utente di interagire con il programma e rendere il tutto più dinamico.

- **MariaDB**: come parte persistente del programma ho scelto di utilizzare una derivazione di MySQL, e come vedremo successivamente ho scelto come tool MariaDB, che possiede una libreria che permette di integrarsi con python e di poter quindi interagire con il database, attraverso il codice, direttamente dal programma.

- **Matplotlib**: è una libreria che permette di creare grafici e immagini statiche, animate e interattive in Python. Nell'ambito di questo progetto, ho quindi utilizzato questo tool per la creazione dei grafici utilizzati negli indici, che poi verranno passati al front-end per essere mostrati a schermo.

2.2 MariaDB

MariaDB nasce nel 2009 da una fork di MySQL, creato dal programmatore finlandese Michael "Monty" Widenius, che ha iniziato e guidato per molti anni il programma originale.
Prende il nome dall'area di sviluppo principale, precedentemente chiamato Maria da cui deriva MariaDB.
Nel dicembre 2012 nasce la MariaDB Foundation con lo scopo di garantire che MariaDB rimanga per sempre un software libero che abbia una buona interoperabilità verso altri programmi.
Negli anni MariaDb ha guadagnato sempre più popolarità tra gli sviluppatori prendendo sempre più il posto di MySQL, come nel caso di Arch Linux e Red Hat.

MariaDB è un DBMS, che sta per database management system, ovvero un software che permette la creazione, manipolazione e interrogazione efficiente di un database.
Ma cos'è un database? In parole povere, un database possiamo vederlo come una libreria, dove ogni box è una tabella, e in esso sono contenute varie informazioni, detti record (righe), suddivisi da dei campi (colonne, dettagli del record).
Si chiamano database relazionali, poiché sono dei database, costituiti dalle varie relazioni interne tra le tabelle, collegamenti (le join).
Questo tipo di struttura, però ha qualche difetto, come quello di avere delle informazioni spezzettate su più tabelle, collegate tra di loro. Questi collegamenti comportano dei rallentamenti, e delle ripetute richieste di query sql al database. Ma le query non sono solo in uscita, ma anche in entrata, proprio perché anche il semplice inserimento, modifica o eliminazione di un dato, comporterà poi un mix di richieste al database relazionale.

La scelta di utilizzare MariaDB come provider per la creazione del Data warehouse è dovuta al fatto che MariaDb essendo open-source permette l'integrazione di diversi linguaggi di programmazione, come nel mio caso con python; ed inoltre è stata una scelta presa in conformità alle policy di Deltatre che utilizza per l'appunto mariadb per altri prodotti e che quindi ne conosce l'utilizzo.

2.3 SQL

SQL è l'acronimo di Structured Query Language, che è uno strumento che permette di fare l'accesso e manipolare i database.

SQL è diventato uno standard dell'American National Standards Institute (ANSI) nel 1986, e dell'International Organization for Standardization (ISO) nel 1987.

Nello specifico SQL può eseguire delle "query" su un database, ovvero delle operazioni su di esso.
Le varie operazioni che si possono effettuare su un database sono:

- inserire un record.
- fare un update dei record.
- cancellare dei records.
- creare nuovi database.
- creare nuove tabelle dentro i database.
- creare delle stored procedures.
- creare delle viste.
- modificare i permessi relativi a tabelle, procedure e viste.
- ottenere i dati da una tabella.

Nell'ambito del mio progetto il linguaggio SQL è stato di fondamentale importanza poiché grazie a questo strumento ho potuto inserire i dati dentro il database, creare tutto il datawarehouse, e in più ho potuto creare le "viste" ovvero delle query precaricate che permettono di gestire la singola vista come se fosse una tabella.

2.4 MongoDB

MongoDB è un programma che si basa su una piattaforma document-oriented. Viene classificato come programma per gestire database NoSQL, infatti MongoDB usa documenti di tipo JSON-like con schemi opzionali.

NoSQL è una alternativa ai tradizionali database relazionali, ed inoltre i database NoSQL sono molto utili per lavorare con un grandi quantità di dati. MongoDB quindi permette di gestire, immagazzinare, e ottenere i dati document-oriented, ovvero, una ricca struttura di dati in grado di contenere matrici e altri documenti.
Ciò significa che è spesso possibile rappresentare in una singola entità, la collezione, un oggetto complesso che se avessimo dovuto rappresentare con database relazionali avrebbero richiesto diverse tabelle.

I vantaggi sono notevoli, leggerezza di calcolo (non vengono eseguite operazioni di aggregazione di dati), non vi sono rischi per l'integrità dei dati, poiché non essendoci dati spezzettati, i dati si trovano tutti nello stesso posto, e vengono recuperati così come sono.
Altro vantaggio è la scalabilità dei dati in maniera orizzontale, e non più verticale.
Probabili svantaggi, invece, possono essere quelli dove vi è un progetto già avviato, e quindi bisogna rimettere mano sul codice.

In genere questo tipo di database viene utilizzato da chi ha delle grosse moli di dati archiviati, su cui lavorare, e deve ripetutamente effettuare richieste al database stesso. Vedesi grosse società gestionali, banche, assicurazioni, o i Social Network come Facebook, Google, Twitter, Pinterest e tanti altri.
Tra i vantaggi si notano sicuramente le performance più alte per i tempi di risposta, infatti nei database non relazionali un elemento contiene tutte le informazioni necessarie e dunque non serve usare i dispendiosi "join" come invece avviene per i database relazionali.
Nel mio progetto ho utilizzato mongodb poiché è la piattaforma su cui l'azienda Deltatre salva i file relativi alle partite, perchè essendo file di notevoli dimensioni, si è reso necessario l'utilizzo di questa struttura per i motivi sopra citati.

Capitolo 3

Stato attuale e Obbiettivi

3.1 Evoluzione dell'analisi

Dall'inizio degli anni 2000, l'analisi delle prestazioni nello sport ha visto una drastica trasformazione, sia nei suoi metodi (incorporando modelli statistici avanzati e nuove strutture analitiche) sia nelle tecnologie (es. sensori di rilevamento e altre apparecchiature di rilevamento). Ciò che è iniziato come annotazioni stenografiche con carta e penna, da allora si è evoluto in sistemi e tecnologie computerizzati avanzati che raccolgono grandi quantità di dati relativi alle prestazioni.

L'aumento delle opportunità finanziarie redditizie nella maggior parte dei principali sport, grazie ai ricavi sempre crescenti degli accordi di trasmissione e all'audience globale in aumento, hanno inevitabilmente alzato la posta in gioco per vincere. Di conseguenza, le organizzazioni sportive si stanno ora rivolgendo ad approcci più scientifici e basati sull'evidenza quando gestiscono le loro istituzioni e sviluppano i loro atleti. Gli standard negli sport d'élite per raggiungere e mantenere il successo vengono continuamente innalzati, esercitando una pressione crescente su club, allenatori e atleti affinché sviluppino strutture di allenamento più efficienti, migliorino i processi di sviluppo degli atleti e acquisiscano una migliore comprensione dei fattori che determinano il successo nei tornei principali.

L'ambiente altamente competitivo con margini in costante diminuzione ha innescato l'emergere dell'analisi delle prestazioni come una funzione dietro le quinte indipendente, ma interdisciplinare, specializzata nella valutazione

oggettiva e più spesso quantitativa delle prestazioni. Questo campo relativamente nuovo mira a supportare gli allenatori nell'identificazione delle aree chiave delle prestazioni che richiedono attenzione, valutando l'efficacia delle prestazioni tattiche e tecniche, nonché i punti di forza e di debolezza delle imminenti avversarie.

Il suo scopo è fornire informazioni valide, accurate e affidabili ad allenatori, giocatori e qualsiasi parte interessata per aumentare le loro conoscenze su una particolare area dello sport. Tradizionalmente, l'analisi delle prestazioni sportive è stata definita come un'attività di analisi osservazionale che va dalla raccolta dei dati fino alla consegna del feedback e mira a migliorare le prestazioni sportive coinvolgendo tutti gli allenatori, i giocatori e gli stessi analisti.

L'osservazione della performance viene effettuata sia dal vivo durante l'evento sportivo che dopo la gara attraverso riprese video e statistiche raccolte. Gli analisti delle prestazioni ora possono essere individuati negli stadi, sia nel box degli allenatori che in una posizione separata e ben visibile all'interno degli spalti, annotando eventi e azioni della partita utilizzando software specializzati, come SportsCode, Dartfish o Nacsport. In questo processo, sviluppano rapporti statistici che possono essere inviati in tempo reale ai dispositivi utilizzati dagli allenatori (ad esempio iPhone o iPad) e visualizzano loro un riepilogo delle metriche chiave delle prestazioni, nonché brevi feed video dei principali punti salienti.

Tuttavia, il tempo aggiuntivo disponibile nell'analisi post-partita consente una valutazione più dettagliata delle prestazioni utilizzando ulteriori fonti di dati complementari. I dati utilizzati durante l'analisi post-partita possono provenire da fonti al di là delle osservazioni dell'analista, come dati qualitativi, sequenze video e persino misurazioni dello sforzo degli atleti, della frequenza cardiaca, dei livelli di lattato nel sangue, dell'accelerazione, della velocità e delle metriche di posizione raccolte tramite dispositivi indossabili. Alcuni di questi dati saranno spesso reperiti all'interno del club, ma anche fonti esterne, come quella dei dati forniti da provider come Opta, sono spesso utilizzate in più sport per integrare i database interni. Anche le sessioni di allenamento sono oggetto di analisi, con un monitoraggio continuo dei giocatori per informare le sessioni di debriefing da parte degli allenatori e aiutare a pianificare quella successiva.[8]

3.2 Cos'è la Performance Analysis

La Performance Analysis consiste nell'insieme dei processi che mirano a consentire l'analisi e la spiegazione accurata di un fenomeno motorio e sportivo. Essa non è legata esclusivamente alla prestazione sportiva, ma sicuramente trova ampio spazio in questo ambito di applicazione.[9]

La Performance Analysis in generale (per esempio può riguardare le prestazioni di un azienda) si basa sull'identificare KPI (Key performance indicators) ovvero "indicatori chiave di prestazione" che forniscono un parametro oggettivo di misurazione e un'indicazione circa alcune variabili della performance che si vuole monitorare.[7]

3.3 Data Analyst e Data Scientist nello sport

In questa nuova frontiera del connubio ormai consolidato fra scienza e sport si sono create in particolare due nuove figure professionali: il Data Analyst e il Data Scientist.

Queste figure sono coinvolte sempre in maggior numero nel lavoro quotidiano di team sportivi a livello professionistico. Qual è dunque la natura del loro lavoro? Innanzitutto entrambi sono deputati alle operazioni di Data Analytics, che si può definire come tutto ciò che riguarda i processi di estrazione ed elaborazione di informazioni a partire da un determinato bacino di dati.[5]

Nello sport nel dettaglio il Data Analyst è la figura professionale preposta alla raccolta e rapida elaborazione dei dati di allenamento e gara, con l'obiettivo di fornire informazioni utili al processo decisionale di coaching. Fondamentali in queste operazioni sono la rilevanza delle informazioni elaborate per la prestazione e la loro validità.
La figura del Data Scientist, ovvero di colui che grazie alle sue conoscenze del processo di allenamento e dell'approccio empirico-scientifico elabora procedure valide e efficaci per sostenere la dinamica degli allenamenti, nasce quindi per garantire validità e rilevanza delle variabili analizzate per la prestazione. Idealmente il compito del Data Scientist è quello di strutturare l'esperienza di allenamento attraverso la validazione delle procedure mentre il Data Analyst

si occupa del loro impiego e della rapidità e sostenibilità del processo.[6]

Questa definizione proposta dal CONI, fa riferimento più che altro a sport dove queste figure professionali incentrano il loro lavoro sul miglioramento della prestazione fisica, con competenze quindi più attinenti alla sfera medico-sportiva. Ci sono però sport, come il Tennis ad esempio, in cui invece Data Analysts e Data Scientists vengono impiegati principalmente per svolgere processi di Match Analysis, per cui le competenze richieste riguardano in particolar modo il mix tra scienze statistiche e conoscenze tecniche e tattiche dello sport in questione.

In ogni caso sono fondamentali per queste figure competenze interdisciplinari, che toccano anche programmazione, machine learning (fondamenti delle tecniche di Intelligenza Artificiale e implementazione delle stesse) e data visualization (che riguarda la rappresentazione grafica dei risultati delle elaborazioni effettuate).

3.4 Tennis e dati

Il tennis è uno degli sport di più antica tradizione e tra quelli maggiormente praticati al mondo: infatti secondo il "Global Tennis Report" redatto dell'ITF, la federazione tennistica internazionale, nel 2019 e basato su un'indagine che comprende i dati di 195 nazioni forniti dalle rispettive federazioni nazionali, il numero di giocatori raggiungerebbe gli 87 milioni, vale a dire l'1,7% della popolazione mondiale.
Gli appassionati di tennis si distribuiscono in 71.263 mila club in 32 nazioni e 489.135 campi da gioco nel mondo: per ogni club, ci sarebbero quindi circa 771 praticanti.[10]

Per quanto riguarda l'accostamento del tennis al mondo della Data Analytics, bisogna cominciare sottolineando che il contesto offre ottime premesse: il tennis infatti è uno sport che ben si presta all'analisi dei dati poiché il sistema di punteggio utilizzato è caratterizzato da unità elementari, i punti, gerarchicamente strutturati in aggregati di punteggio discreti (game e set) e inoltre l'esito finale dei match è binario (vittoria o sconfitta).
Paradossalmente però i dati solitamente disponibili pubblicamente sono incentrati sugli aggregati di punteggio elementari (punti, game e set vinti),

i singoli "punti importanti" vinti (i cosiddetti "break points") e indicatori di performance non troppo approfonditi sull'unico colpo dettagliatamente tracciato (il servizio), come ace e doppi falli.
Oltretutto per quanto riguarda le statistiche offerte ai fruitori del prodotto televisivo (i proventi derivati dai diritti tv sono strategici per quanto riguarda l'economicità dei circuiti professionistici), che dovrebbero contribuire ad elevarne la qualità, bisogna evidenziare che ci sono importanti margini di miglioramento, se si considerano le potenzialità di questo sport.

Ad ogni modo, nonostante un evidente ritardo se comparato ad altri sport, specialmente quelli appartenenti al mondo del professionismo americano (Basket e Baseball in primo luogo), anche il Tennis sta cercando di compiere passi in avanti verso un approccio sempre più rivolto alle filosofie data-driven, anche tramite l'inserimento delle figure professionali citate in precedenza nei teams degli atleti.
L'uso di tecniche di analisi dei dati infatti sta diventando sempre più consueto fra i top player e sta dando vita a un cambiamento anche culturale, richiedendo la presenza fissa all'interno dei team degli esperti di analisi dei dati che possano in primo luogo analizzare le grandi quantità di dati potenzialmente a disposizione e riuscire inoltre ad offrire ai coach informazioni ed elaborazione di facile lettura.
A partire da queste informazioni, il team tecnico di un atleta può costruire sia i piani tattici per i singoli match (la strategia), quanto sviluppare piani di allenamento che, nel lungo periodo, possono modificare e migliorare il suo modo di giocare.

Nel circuito professionistico femminile gli strumenti di analisi dei dati raccolti nei tornei sono attualmente forniti da SAP [11] e consentono però ai soli allenatori di accedere temporaneamente alle informazioni più dettagliate sull'andamento dei match in tempo reale.

Nel circuito maschile invece un altro colosso dell'Information Technology come Infosys ha recentemente rinnovato l'accordo di collaborazione con l'ente che gestisce i tornei in qualità di Global Technology Services Partner and Digital Innovation Partner fino al 2023.[12]

3.5 Deltatre

3.5.1 Chi è?

Deltatre, società italiana fondata a Torino nel 1986, ma ormai con basi operative dislocate in tutto il mondo.
Offre una vasta rosa di servizi, segue i più grandi eventi sportivi generalisti, come le Olimpiadi, ma anche tornei, campionati e leghe di calcio, football, golf, nuoto, rugby e tennis.

Durante l'intervallo o alla fine di una partita, tutti noi guardiamo più o meno attentamente le statistiche che vengono presentate, per poterci fare una idea del motivo di un determinato risultato.
Dietro questi numeri non ci sono solo grandi computer che elaborano i dati, ma c'è un team di persone che applica modelli, esegue conteggi e verifica tutto ciò che noi poi vediamo racchiuso in statistiche o in grafiche.

Il loro lavoro non si limita solamente al controllo analitico dei dati, ma forniscono una esperienza a tutto tondo che riguarda sia i servizi legati all'intrattenimento, ovvero programmi trasmessi dalle piattaforme internet e in televisioni, sia sistemi di interazione con il pubblico come interfacce, grafiche, siti, applicazioni, clip video.
Inoltre, forniscono sistemi di supporto per telecronisti e arbitri, come accade nel caso del calcio con il Var, dove l'azienda non gestisce direttamente lo strumento d controllo ,ma si occupa della gestione delle immagini in tv e sui monitor, al fine di mostrare in diretta quello che sta accadendo in campo e fornendo un maggiore grado di coinvolgimento per lo spettatore.

In ogni partita, torneo o altro evento sportivo all'interno del team collaborano diverse persone, ognuna con un compito diverso, tra cui troviamo ad esempio un operatore che, con l'ausilio del tracking, registra ogni azione(tiri, passaggi, battute, etc.) compiuta da ciascun giocatore in campo, mentre un'altra persona invece si occupa di elaborare contenuti, ad esempio clip video, pronti ed impacchettati da offrire ai media, i quali diventano poi gli highlights che vengono poi mostrati in diretta.

Inoltre vi è tutto il team di analisti e matematici, che una volta concluso il match si occupa di eleborare i dati raccolti e studiare dei modelli per poter ottenere le grafiche e le statistiche mostrate sui vari media.
Lo stesso lavoro viene compiuto dall'azienda nell'ambito del Tennis, motivo per cui nel momento in cui mi si è reso necessario ottenere dei dati e delle informazioni mi sono rivolto a loro per poter procedere nello svolgimento della tesi.

3.5.2 Come raccoglie i dati oggi

Deltatre collabora al giorno d'oggi con le maggiori competizioni tennistiche in tutto il mondo, dove gli analisti lavorano per poter collezionare i dati relativi ai match.
Le informazioni raccolte per ogni punto si presentano su tre livelli diversi di dettaglio, il primo riguarda le informazioni che il giudice di sedia immette nel suo palmare: il tempo di gioco, i servizi sbagliati, il punteggio.
Poi ci sono i dati inseriti manualmente in tempo reale, come chi ha vinto il punto, se era un dritto o un rovescio, se c'è stato un errore e se è stato forzato o meno.
Infine vi è un terzo livello di dati, rilevati automaticamente dalle attrezzature tecnologiche che sorvegliano il campo, come la velocità del servizio o il numero di palleggi dello scambio. Si tratta di migliaia di dati ogni partita.
Come abbiamo detto prima vi sono dati più oggettivi e facili da raccogliere, che vengono collezionati direttamente dall'azienda, mentre quelli più complessi, come quelli di terzo livello, vengono forniti da provider specializzati di terze parti che aiutano ad avere una visione completa del dato.[13]

3.6 Come vengono utilizzati i dati oggi

Ma come vengono utilizzati i dati collezionati oggi? Lo scopo principale è sempre quello informativo, ovvero informare l'utenza sull'andamento della partita in corso o conclusa, oppure per mostrare lo stato di forma del giocatore, mostrando lo storico recente delle partite.

Nelle figure 3.1, 3.2 possiamo notare degli esempi di grafiche che riportano le informazioni sopra citate.

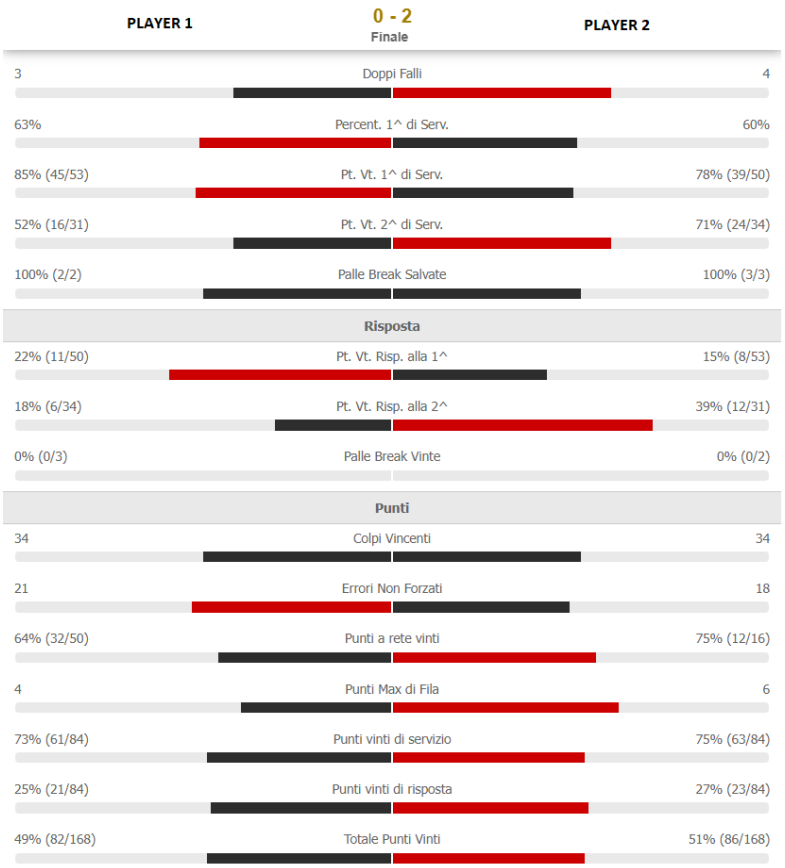

Figura 3.1: esempio di infografica di un match di tennis

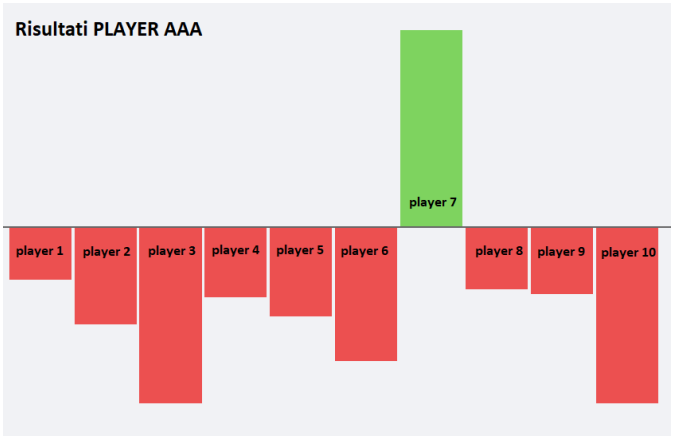

Figura 3.2: esempio di infografica di riepilogo forma di un giocatore

Come possiamo evincere dalle due figure le grafiche odierne sono piuttosto scarne e poco esplicative, difatti questi dati agli occhi dei meno esperti non riescono a descrivere appieno il controllo della partita, ovvero se un giocatore ha dominato il match o se è stato equilibrato. Ma neanche lo stato di forma di un giocatore, infatti esso va valutato considerando più aspetti.

Per questo con Deltatre si è pensato di costruire un sistema che permetta di descrivere più facilmente le partite, andando a creare metriche e indicatori che possano meglio rappresentare il dominio, o meno, del match. Allo stesso tempo queste nuove tipologie di dato si possono anche utilizzare per poter fare confronti tra giocatori andandoli a valutare all'interno dei loro fondamentali, come servizio, risposta, tie-break, volee a rete, etc.

Capitolo 4

Tipologia e struttura dati

4.1 Tipologie di dati

Per il mio studio sono partito da una serie di file json provenienti da un server mongoDB di un provider dell'azienda con cui ho collaborato. I file che vengono forniti dal provider si dividono in 3 categorie, i quali vengono forniti tutti in formato documentazionale "json":

- *Umpaired Data*, ovvero i dati provenienti dal seggiolino dell'arbitro e che vengono registrati in tempo reale dal tablet che si trova sul posto, questi registrano il punteggio e chi ha fatto il punto, quindi con un livello di dettaglio basso.
- *Logger Data Level2*, che sono i file prodotti dagli operatori presenti alla partita e che registrano i punti fatti dai giocatori, ma inoltre segnalano anche la tipologia del punto e come è stato prodotto, aumentando il livello di dettaglio del dato.
- *Logger Data Level3*, che sono i dati più tecnici e più dettagliati, come il tracking del giocatore o della pallina o la velocità della pallina, che vengono forniti da aziende specializzate nel settore.

Nel mio caso mi sono stati forniti i file di livello1 e livello2, ma nell'ambito della mia ricerca ho utilizzato principalmente i json di secondo livello, che hanno maggiori informazioni utili per lo sviluppo del mio progetto. I file json sono un tipo di documento che attraverso delle librerie di codice è facilmente navigabile, poiché ben formattato.

Nel caso dei file trattati in questo progetto i punti del match vengono registrati punto per punto in maniera incrementale, e anche le variabili relative alla tipologia di punto sono rappresentate da un contatore che viene incrementato puntualmente.

4.2 Tipologia di variabili

Abbiamo quindi detto che nell'ambito della ricerca ho analizzato i file di livello2, quindi i file con le informazioni inserite dagli scout sulla tipologia di colpo o sulla tipologia di punto.
Ma in che modo un giocatore può fare punto? Nel tennis l'obbiettivo di un giocatore è mandare la pallina nell'altro campo al di sopra della rete colpendola con la racchetta. Per poter acquisire un punto il giocatore deve mettere in difficoltà l'avversario cercando di far compiere alla pallina due rimbalzi nel campo avversario, o forzandogli l'errore.[14]

Il gioco del tennis si basa su dei colpi chiamati "fondamentali", che vengono insegnati fin da subito nelle scuole tennis, e sono diritto, rovescio e servizio. Inoltre all'interno del gioco effettuando delle variazioni all'impugnatura e modificando il metodo di realizzazione dei colpi sopra citati si possono effettuare altri tipologie di colpi come la volée, il lob, lo smash e la palla corta.[15]

4.2.1 Fondamentali

Nel ***servizio***, affinché vada a buon fine e possa dare avvio allo scambio, la pallina deve superare la rete e poi rimbalzare nel campo di battuta opposto, diagonalmente a quello dal quale si batte.
Il giocatore si alza la palla in alto e la colpisce nel suo punto massimo, così da ottenere la miglior traiettoria possibile.
è un fondamentale molto importante, in quanto consente, se ben realizzato, di dominare da subito lo scambio. Inoltre se la battuta è ben eseguita e forte, il giocatore guadagna un punto diretto chiamato ***ace***.

Il giocatore al servizio però potrebbe commettere un errore mandando la pallina fuori dal campo o in rete, in questo caso si dice che il giocatore al servizio commette ***fallo***, per cui la battuta deve essere rigiocata, nel caso in

cui il battitore commetta due "falli" consecutivi concede un punto diretto all'avversario poichè commette il ***doppio fallo***.

Il ***diritto***(Forehand) è il colpo di tennis per eccellenza, il fondamentale più facile e probabilmente il primo colpo che si impara nella scuole tennis, si tratta di quel colpo che si effettua quando la palla viene colpita alla propria destra (o sinistra per i giocatori mancini) dopo il rimbalzo.
A seconda della superficie su cui viene giocato il match questa può modificare il modo in cui rimbalza la pallina creando qualche problematica, ma per il giocatore rimane comunque forse il colpo più importante e maggiormente allenato.
La maggior parte dei tennisti usano una sola mano per afferrare la racchetta per il dritto, ma alcuni utilizzano un'impugnatura a due mani.

Il ***rovescio***(Backhand) è quel colpo che si effettua quando la palla viene colpita alla propria sinistra (o destra per i giocatori mancini) dopo il rimbalzo.
Il rovescio potrebbe essere eseguito a una mano o a due mani, ciascuna tipologia presenta dei vantaggi e degli svantaggi al giocatore, in linea di massima, il colpo a due mani è più efficace nel gioco di rimbalzo, in quanto comporta un movimento più corto e permette una maggiore accelerazione nella risposta, mentre il rovescio ad una mano è particolarmente favorevole nel gioco al volo.

La ***volée***(Volley) rappresenta uno tra i colpi più spettacolari ed apprezzati dal pubblico.
E' un colpo aggressivo che viene normalmente prodotto per forzare l'avversario o per vincere un punto a rete: in generale si tratta del colpo in cui la palla viene colpita prima che rimbalzi, con l'obbiettivo di di spedire la palla nel capo avversario prima che l'avversario abbia il tempo di reazione necessario.
Vi sono diverse tipologie di volée, oltre alla classica di dritto e rovescio, come quella smorzata, la bloccata, la volée lob e la mezza volée.
In passato i giocatori effettuavano il cosidetto ***serve & volley***, attraverso il quale un giocatore si spinge a rete per poi rispedire rapidamente la palla al suo avversario.

4.2.2 Altre tipologie di colpi

Il ***lob*** è un colpo del tennis che richiede abilità, controllo e tecnica, e consiste in un pallonetto che può avere finalità difensiva o di attacco.
Il lob di difesa si usa quando si vuole rallentare il gioco perché si è in difficoltà, sia perché si vuole guadagnare tempo utile a recuperare, oppure perché si è fuori posizione e la si vuole recuperare. Inoltre potrebbe venire usato anche per spezzare il ritmo e indurlo all'errore. Invece il lob offensivo si esegue per mettere in difficoltà l'avversario che si trova già a rete, costringendolo a rincorrere la palla all'indietro con un pallonetto, costringendolo ad una corsa innaturale e difficoltosa.

Lo ***smash*** è, in parole semplici, il colpo di risposta ad un pallonetto. Può essere realizzato al volo o in seguito ad un rimbalzo, e solitamente porta ad un punto diretto.
Il movimento che viene eseguito è analogo a quello che si effettua nel caso nel servizio, ma con la differenza che la traiettoria della palla dipende dal precedente colpo dell'avversario.

Il ***drop shot*** è spesso un colpo risolutivo, tra i più difficili e spettacolari del tennis. E' un colpo che, tramite taglio ed effetto sulla palla, richiama l'avversario a rete.
È un colpo che si esegue soprattutto quando l'avversario è distante o in posizione sfavorevole, in modo che arrivi con difficoltà a raggiungere la palla.

Lo ***stroke net*** è un colpo in cui il giocatore forza il colpo andando a colpire il nastro superiore della rete, che nell'impatto smorza la velocità della pallina, facendola cadere nel campo avversario.

Il ***drive*** è un colpo molto comune ed è quando il giocatore colpisce la palla di dritto, ovvero quando si trova sul suo braccio più forte.

Il ***pass*** è un altro colpo del tennis spettacolare e spesso risolutivo, eseguito per superare l'avversario quando questi si porta sotto rete. Il passante può essere "lungolinea", quando la palla viene scagliata parallelamente alla linea laterale, oppure "incrociato", quando la traiettoria attraversa il campo in diagonale.

4.3 Contenuto Dataset

Il file json si presenta come un vettore di singoli blocchi che rappresenta ognuno un punto giocato. Ciascuno di questi blocchi possiede alcune variabili globali che rimangono invariate, come gli header e le informazioni relative agli strumenti e le librerie utilizzate per creare i file, ed inoltre vi sono delle variabili proprie del mongoDB, quali l'id del punto, il timestamp di creazione e di ultima modifica.

Il "blocco punto" si articola poi in diverse sezioni sotto la voce "Content", che racchiude il dato vero e proprio.
Nella prima parte della sezione possiamo trovare le seguenti voci relative alle informazioni tecniche e genrali riguardanti il match:

- *Year*: rappresentato da un intero definisce la stagione in corso in cui si sta giocando il match.
- *TournamentId*: codice numerico che definisce univocamente il torneo in cui si sta giocando il match.
- *MatchId*: codice alfanumerico che rappresenta univocamente il match all'intero del torneo e allo stesso tempo la tipologia di partita in corso. (es Quarto di finale, Semi finale, etc.)
- *CourtId*: codice alfanumerico che indica su quale campo della struttura si sta giocando il match.
- *Timestamp*: è un timestamp in formato UTC che indica l'orario in cui viene registrato il punto.
- *DayNumber*: indica il giorno corrente relativo al torneo in corso.
- *Date*: campo in formato UTC che indicata la data del giorno del match.
- *Duration*: indicatore progressivo, in formato day:hour:minutes:seconds .milliseconds, della durata effettiva del match.
- *StartTime*: indica l'orario in cui è iniziato il match.

Successivamente troviamo una serie di informazioni relative alle informazioni anagrafiche dei due team partecipanti, alcune informazioni relative al punteggio e al servizio.
Questa sezione di codice si ripete per ciascun team:

TeamOne/Two: intestazione del sottoinsieme di dati, che si ripete per ciascun team, riguardanti il giocatore, o i giocatori in caso di doppio.

- PlayerOne/Two: sotto insieme che mostra le anagrafiche del giocatore, o giocatori.
  Questa sezione viene ripetuta per entrambi i giocatori:
  - DisplayName: Stringa che rappresenta il cognome del giocatore, viene chiamato "display name" perchè è il nome con cui viene rappresentato sul tabellone e sullo schermo.
  - FirstName: Stringa che rappresenta il nome del giocatore.
  - Number: Codice alfanumerico identificativo con cui viene contrassegnato il giocatore in modo univoco.
  - CountryCode: Codice alfabetico univoco che indica la nazionalità del giocatore.
  - Score: Sezione che descrive il punteggio del team all'interno del match.
    Si articola in diversi sottocampi, infatti in questa sezione troviamo il punteggio del giocatore all'interno del game (15-30-40-Ad), ma anche i punteggi dei set e dei tie break giocati dal relativo giocatore.
- IsServer: booleano che indica se il team contrassegnato sta servendo.
- IsNextServer: booleano che indica se il team contrassegnato sarà il prossimo a servire.
- IsRetired: booleano che indica se il team contrassegnato si è ritirato dal match.
- IsWinner: booleano che indica se il team contrassegnato ha vinto il match, generalmente viene aggiornato con l'ultimo "blocco punto".

In nella sezione seguente troviamo invece le informazioni realtive colpi e alle tipologie di punto, esse tengono il conteggio totale dei relativi colpi che si sono verificati fino a quel momento, ad esempio nel caso in cui si verifichi un ace, il contatore degli ace si andrebbe ad incrementare di una unità. Anche in questo caso abbiamo un blocco intero per ogni team:

Team One/Two Total: intestazione del sottoinsieme dei dati realativi alle tipologie di punto.

- Win Forehand/Backhand: variabile che mantiene il conteggio totale dei colpi vincenti di dritto/rovescio.
- Forced Forehand/Backhand: variabile che mantiene il conteggio totale degli errori forzati commessi di dritto/rovescio.
- Unforced Forehand/Backhand: variabile che mantiene il conteggio totale degli errori non forzati commessi di dritto/rovescio.
- Service Winner: variabile che mantiene il conteggio totale dei servizi che hanno portato ad un punto diretto che però non è un Ace.
- Net Points Won: variabile che mantiene il conteggio totale dei punti vinti a rete.
- Baseline: variabile che mantiene il conteggio totale dei punti vinti da fondo campo.
- Ace: variabile che mantiene il conteggio totale degli ace fatti dal giocatore.
- DoubleFault: variabile che mantiene il conteggio totale dei doppi falli commessi dal giocatore.
- Service Played: variabile che mantiene il conteggio totale dei servizi giocati.
- Return Played: variabile che mantiene il conteggio totale delle risposte giocate.
- First Serve: variabile che mantiene il conteggio totale dei primi servizi giocati.

- Point On First/Second Serve: variabile che mantiene il conteggio totale dei punti fatti sui primi/secondi servizi.
- Break Point Saved: variabile che mantiene il conteggio totale dei break point salvati.
- Break Point Won: variabile che mantiene il conteggio totale dei break point vinti.
- Point On First/Second Return: variabile che mantiene il conteggio totale dei punti fatti in risposta ad un primo/secondo servizio.
- Total Service Won: variabile che mantiene il conteggio totale dei servizi vincenti.
- Total Return Won: variabile che mantiene il conteggio totale delle risposte vincenti.
- Total Point Won: variabile che mantiene il conteggio totale dei punti vinti.
- Shot: sotto insieme che raggruppa le informazioni relative ai colpi
  - Type Name: (Dropshot, Lob, Drive, Pass, Passing Volley, Return, Smash, Stroke net, Volley)
  - Win: numero di colpi vincenti legati alla relativa tipologia di punto
  - Win Backhand/Forehand numero di colpi vincenti di dritto/rovescio legati alla relativa tipologia di punto
  - Error numero di errori legati alla relativa tipologia di punto
  - Forced Forehand/Backhand numero di errori forzati di dritto/rovescio legati alla relativa tipologia di punto
  - Unforced Forehand/Backhand numero di errori non forzati di dritto/rovescio legati alla relativa tipologia di punto

Team One/Two Sets: sezione che raccoglie i dati sulla tipologia di punto e colpo con cui vengono conquistati, raggruppando per set, anche in questo caso abbiamo tutte le voci di cui sopra.

Capitolo 5

Progettazione del datawarehouse e sviluppo dell'applicazione

5.1 Definizione del problema

L'obbiettivo di questo progetto è quello di creare una piattaforma che permetta in modo semplice e intuitivo di ottenere in tempo reale delle metriche e indicatori che permettano di misurare le performance di un giocatore o poter metterne a confronto due.

I primi passi da fare sono quelli di capire come, partendo dal dataset presentato in precedenza, estrarre e gestire i dati provenienti dai file json.

I dati andranno salvati su un database, per cui si rende necessario la progettazione di un datawarehouse che gestisca tutti i dati provenienti dai file, e li "intersechi" con le tabelle accessorie che andranno a comporre il nostro database.
Per poi poter fare un upload veloce dei dati sul nostro database il formato più comodo è quello di fare delle insert partendo da un file CSV, per cui il secondo obbiettivo è quello di andare a costruire un script che processi i vari file e li trasformi in un file csv.

Il passo successivo dopo aver immagazzinato i dati, è quello di fare le interrogazioni così da avere le informazioni base per poter iniziare a costruire

le metriche e gli indicatori.

Una volta studiati e costruiti i KPI e le metriche che vogliamo ottenere, è necessario automatizzare il tutto e costruire uno script che esegue tutti i passaggi in fase di esecuzione.

L'ultimo passo è quello di "confezionare" il tutto dentro una applicazione semplice da utilizzare e che mostri gli indici e le grafiche prodotte.

5.2 Pre-Processamento dei dati

Come anticipato nel capitolo precedente, il nostro punto di partenza è stato un file json.
Purtroppo i file forniti non erano in un formato json standard per cui non potevano essere letti in maniera documentazionale utilizzando la libreria integrata in python, quindi si è reso necessario fare un ulteriore step prima di poterli utilizzare, andando a processarli per renderli idonei al formato standard.
Il problema risiedeva in una versione obsoleta di MongoDB usata per esportare i dati, per cui la soluzione che abbiamo trovato è quella di importare nuovamente i file, utilizzando la "Legacy mode", su un mongoDB temporaneo e riesportarli in un formato json leggibile per poter procedere all'estrazione.

Le operazioni di import export si possono effettuare anche utilizzando il terminale, per cui utilizzando python, il quale permette di invocare comandi sul terminale ho deciso di automatizzare questo processo utilizzando un script, che, iterando dentro una cartella, prende ogni file e lo importa ed esporta in automatico, eseguendo i seguenti comandi:

- mongoimport –db Tennis –collection pointByPoint –legacy –drop –file ./path/to/file
- mongoexport –db Tennis –collection pointByPoint –pretty –jsonArray –out ./path/to/file

Di seguito le operazioni descritte in precedenza in pseudo-codice:

```
for elem in glob.glob(path+'\*'):
    cmd = '''%s -db Tennis -legacy -collection pointByPoint -drop -file %s'''
                                                                % (import_path, elem)
    os.system(cmd)
    print('''%s importato con successo'''%(elem))
    list = elem.split('\\')
    name = list[len(list)-1]
    list.remove(name)
    filename = ( 'extracted_%s' % (name) )
    list.append('extracted')
    list.append(filename)
    new_name = '\\'.join(list)
    os.system('''%s -db Tennis -collection pointByPoint -pretty -jsonArray -out %s'''
                                                                % (export_path,new_name))
    print('''%s esportato con successo'''%(new_name))
```

5.3 Estrazione dei dati, da file Json a CSV

Una volta ottenuti i file json leggibili, ho proceduto alla fase di estrazione, per estrarre i dati ho utilizzato sempre uno script python che itera su tutti i file json pre-processati e per ciascuno di essi elabora l'estrazione.
Per compiere le operazioni e interpretare i file json ho utilizzato la libreria json fornita da python, l'approccio che ho utilizzato quindi è stato quello di caricare tutto il file in una variabile, come se facessi una lettura su file.
Come abbiamo visto prima i file si presentano come un vettore di blocchi, per cui ho iterato sui singoli blocchi punto e ho, per ciascuno di essi, compiuto le operazioni di estrazione.
Per estrarre il dato relativo al singolo punto e quindi associare il punto alla tipologia e alla modalità con cui vengono fatti, visto che si tratta di contatori progressivi ho salvato per ogni iterazione il punto "corrente" e il blocco dato successivo, che corrisponde in ordine cronologico al punto "precedente", e per associare quindi l'informazione al dato ho effettuato un confronto per differenza andando ad associare al giocatore che ha "cambiato punteggio" l'informazione dove cambiano i contatori.
Per alcune opzioni abbiamo l'informazione sia per il team1 che per il team2 per cui per completezza ho salvato tutte le informazioni per entrambi giocatori.

In pseudo-codice lo svolgimento avviene in questa maniera:

```
for point in data:
    CurrentPoint = data[point]
    PreviousPoint = data[point + 1]
    for pidx in ['TeamOneTotal', 'TeamTwoTotal']:
        for type in typeofpoint:
            if not isinstance(CurrentPoint['Content'][pidx][type], int):
                if CurrentPoint['Content'][pidx][type]['Count'] !=
                                                PreviousPoint['Content'][pidx][type]['Count']:
                    description = description + type + '-Count' + " "

                if CurrentPoint['Content'][pidx][type]['Total'] !=
                                                PreviousPoint['Content'][pidx][type]['Total']:
                    description = description + type + '-Total' + " "
            else:
                if CurrentPoint['Content'][pidx][type] !=
                                                    PreviousPoint['Content'][pidx][type]:
                    description = description + type + " "

        currshots = CurrentPoint['Content'][pidx]['Shots']
        prevshots = PreviousPoint['Content'][pidx]

    stringtowrite = ( 'time %s|%s| scores: %s - %s|
        Set1: %d - %d Tie: %d - %d| Set2: %d - %d Tie: %d - %d|
        Set3: %d - %d Tie: %d - %d| other %s| Type of point: %s \n'
        % (timestamp, pname, scorep1, scorep2, p1set1, p2set1, p1tie1,
        p2tie1, p1set2, p2set2, p1tie2, p2tie2, p1set3, p2set3, p1tie3,
        p2tie3, other_description, description))
```

Figura 5.1: figura che mostra le differenze tra blocchi, a destra abbiamo un pezzo del file json riguardante il punto "corrente" mentre a sinistra lo stesso pezzo, ma riguardante il punto "precedente"

Un esempio di come ho lavorato possiamo evincerlo dalla figura 5.1, dove possiamo vedere la stessa parte di "blocco punto", ma in due punti successivi,

per cui come nella figura sono andato a vedere dove il dato si incrementava, e ho associato l'informazione al giocatore, infatti nell'esempio possiamo notare che si è incrementata la voce "forced" e "forced forehand" del sotto-insieme "TeamOneTotal", per cui il giocatore 1 ha commesso un errore non forzato giocando sul rovescio, inoltre possiamo anche vedere che il numero totali di "Baseline" è aumentato per cui il giocatore in quel momento si trovava a fondocampo e infine il fatto che il numero totali di "unreturn serve" è aumentato, indica che il giocatore stava servendo e che quindi il dato statistico relativo al numero di servizi rappresentato da "total" è stato incrementato.
Una volta valutate tutte le differenze, per comodità ho scritto una stringa che esprimesse quindi tutte le informazioni, creando un file.txt per ogni match, per ogni file json analizzato.
La stringa scritta ha il seguente formato:

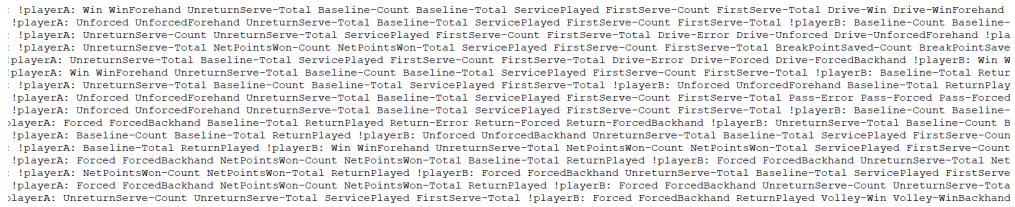

Figura 5.2: esempio di file .txt prodotto dall'estrazione

Queste stringhe vanno lette, per esempio prendendo la prima, nel seguente modo: «il player A ha vinto il punto con un dritto, stava servendo lui, e ha vinto giocando da fondo campo, facendo un colpo di tipo "Drive"».
Elaborate tutte le differenze e quindi estratti i dati dal file json, ho cercato di creare qualche dato in più sulla base di quelli estratti, definire se si trattasse di un break point, o di tie break, etc. Una volta ottenuti tutti i dati utili, ho creato un file csv che riassumesse il tutto affinché fosse facile importarlo su database in quanto un file csv ha di fatto già un formato tabellare.
Inoltre questo step mi è stato utile anche per poter indicizzare tutti i dati andando a sostituire i termini "espliciti" con i relativi codici delle tabelle esterne, così facendo ho potuto caricare sulla tabella principale i dati già pronti per poter fare le join le altre tabelle. Per poter fare tutto ciò ho dovuto creare dei dataframe che contenessero le informazioni esterne, che poi ho utilizzato per poter fare i "match" con i dati per ottenre gli identificativi. Per iniziare, quindi, ho stabilito una connessione con il database, ho scaricato le tabelle esterne che mi servivano e le ho salvate in dei DataFrame, tutto

questo in real-time durante l'esecuzione.

```
# Get Cursor
cursor = conn.cursor()
dictionary = {}

for database in list_of_database:
    sql_select_Query = "select * from %s" %(database)
    cursor.execute(sql_select_Query)
    # get all records
    records = cursor.fetchall()
    dictionary[database] = records

conn.close()

match_dict = dictionary['match']
competition_dict = dictionary['competition']
player_dict = dictionary['player']

for line in competition_dict:
   if tournamentId in line:
       tournamentId = line[0]
       break

for line in match_dict:
   if year and matchId in line:
       if line[2]==tournamentId and line[3]==year:
           matchId = line[0]
           break

for line in player_dict:
    if p1name in line:
        p1Code = line[0]
    if p2name in line:
        p2Code = line[0]
```

5.4 Creazione del Datawarehouse

Per la creazione del Datawarehouse ho utilizzato come detto prima MariaDB, che ci ha permesso di utilizzare un tool molto utile come HeidiSQL. La tabella pointByPoint utilizzata come centro stella del nostro datawarehouse possiede i seguenti attributi relativi al match a cui appartiene:

- *pointByPoint_id*: identificativo univoco del punto.
- *timestamp*: timestamp di quando è stato fatto il punto.
- *competition_id*: identificativo del torneo facente riferimento alla tabella "Competition".
- *year*: anno in cui viene giocato il torneo.
- *match_id*: identificativo del match relativo alla tabella "match" del datawarehouse.
- *duration*: durata effettiva del punto.

Poi ci sono una serie di informazioni relative ai giocatori e altre informazioni accessorie che arricchiscono le informazioni sulla tipologia di punto:

- *playerA_id*: codice intero che identifica univocamente il giocatore A all'interno della tabella "player".
- *playerB_id*: codice intero che identifica univocamente il giocatore B all'interno della tabella "player".
- *currentGame*: intero progressivo che indica il numero del game corrente all'interno del set.
  Questo campo può assumere valori da 1 a 13 nel caso di tiebreak.
- *currentSet*: intero progressivo che indica il set corrente all'interno del match.
  Questo campo può assumere valori da 1 a 12 nel caso di tiebreak.
- *gameType*: intero che indica la tipologia di game tra: normale, tiebreak, match tiebreak.
  Viene calcolato utilizzando le informazioni contenute nei campi "currentSet" e "currentGame".
- *isBreakPoint*: booleano che indica se il punto relativo è un break point o no.
- *isGameSetMatchPoint*: intero che descrive la tipologia di game in cui si sta svolgendo il punto tra: normale, game, set e match point.
  Viene calcolato utilizzando le informazioni contenute nei campi "currentSet" e "currentGame".
- *scoredBy*: campo che contiene il codice identificativo del giocatore che ha fatto il punto.
- *servingPlayer*: campo che contiene il codice identificativo del giocatore che serve.
- *nextServingPlayer*: campo che contiene il codice identificativo del giocatore che sarà prossimo a servire.

Poi vi sono una serie di informazioni sul punto, come la posizione dei giocatori o i colpi che hanno usato per effettuarli:

- *isFirstServe*: intero che indica se si tratta di una prima o seconda al servizio.
- *isAce*: intero che indica se il punto effettuato è un ace.
- *currentGameScore_A/B*: intero che indica, per ciascun giocatore, il punteggio all'interno del game o di tiebreak.
  In quuesto campo i valori sono 15-30-40 e 45 in caso di "vantaggio".
- *setScore_A/B*: intero che indica, per ciascun giocatore, il punteggio dei set all'interno della partita.
- *tieScore_A/B*: intero che indica, per ciascun giocatore, il punteggio dei tie-break all'interno della partita.
- *shot_A/B*: codice intero che indica, in modo univoco, facendo riferimento alla tabella "Type shot", per ogni giocatore, che colpo ha utilizzato per quel punto.
- *position_A/B*: indica per ogni giocatore in che posizione si trovava in quel punto.
  Questo campo può assumere valore 0 nel caso in cui il giocatore si trovi a fondocampo, oppure 1 nel caso in cui si trovi a rete.
- *winType_A/B*: codice intero che indica, in modo univoco, facendo riferimento alla tabella "Type win point", per ogni giocatore, in che modo ha vinto quel punto.
- *errorType_A/B*: codice intero che indica, in modo univoco, facendo riferimento alla tabella "Type error point", per ogni giocatore, che tipo di errore ha commeso in quel punto.

Come possiamo vedere negli attributi della tabella pointByPoint, possiamo notare che ci sono molti riferimenti a tabelle esterne, infatti in un datawarehouse ottimale il centro stella deve essere codificato così che l'informazione pesi meno e occupi meno spazio, visto che dovrà contenere un alto numero di record, non andando ad inficiare troppo le performance.

Quindi per ricreare l'informazione completa, si deve fare uso di tabelle esterne che espandono l'informazione relativa ad un attributo, tramite l'utilizzo di "join".
Nel nostro caso si è reso necessario al fine della costruzione del datawarehouse la creazione delle seguenti tabelle:

- *Tour*: tabella che registra le informazioni sui tour, ovvero i circuiti e le organizzazioni che organizzano le competizioni.
- *Competition*: tabella che immagazzina le informazioni sulle competizioni, come nome, superficie, tipologia di competizione, luogo di svolgimento, etc.
- *Tournament*: tabella che contiene le informazioni sul torneo, più nello specifico delle singole edizioni dei tornei che ogni anno ciclicamente si ripetono.
- *Match*: tabella che registra le informazioni relative ai match che sono stati giocati, come i giocatori, il torneo, il punteggio finale, il campo, il meteo, etc.
- *Player*: tabella che registra le informazioni anagrafiche relative ai player, come ad esempio nazionalità, mano dominante, età, etc.
- *Ranking*: tabella che registra i ranking e le posizioni dei giocatori nel tempo, salvando per ogni settimana la posizione in classifica del giocatore.
- *Type error point*: tabella che contiene le tipologie di errori che possono essere commessi, come ad esempio se di dritto o rovescio, oppure se errore forzato.
- *Type shot*: tabella che descrive le tipologie di colpo che un giocatore può utilizzare, che sono quelle che abbiamo descritto prima (Lob, Drive, pass, etc.).
- *Type win point*: tabelle che rappresenta le tipologie di punti vincenti di un giocatore, come ad esempio se un errore diretto al servizio, oppure se di dritto o rovescio, etc.

Il datawarehouse si presenta quindi con la forma che possiamo vedere nella Figura 5.3.

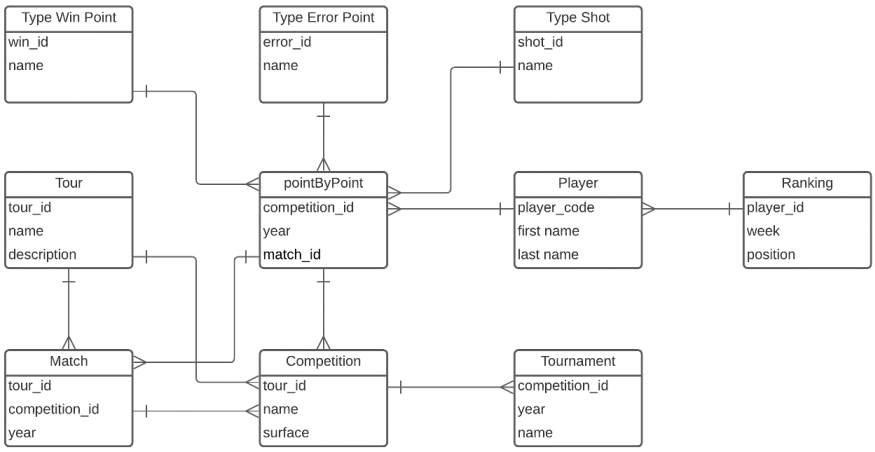

Figura 5.3: struttura e grafico del datawarehouse

Queste tabelle esterne sono stato anch'esse create nell'ambito del nostro progetto, ma potrebbero essere database forniti da provider esterni che tengono aggiornate in maniera autonoma le informazioni e che possono esssere facilmente integrate con il nostro progetto.

5.5 Creazione dei Dati Raw attraverso le viste SQL

Per poter iniziare a creare indici, metriche e statistiche abbiamo bisogno di dati "Raw", ovvero dati "grezzi" che, una volta aggregati, permettono di poter ottenere le metriche e i KPIs.
Per poter ottenere i dati Raw ho utilizzato SQL che è necessario per interrogare i databases. Il risultato di queste query sql non sono altro che altre tabelle che andranno analizzate a loro volta.
Per poter avere i dati base necessari, ho dovuto crearmi delle tabelle custom che mi permettano di prepare i dati per le query. Ho utilizzato un strumento molto utile come le "viste", ovvero delle query precaricate che permettono di essere trattate e interrogate come se fossero dei database a sé.

Nella nostra analisi abbiamo utilizzato diverse query SQL, di seguito le viste create:

- **Numero di ace**

```
SELECT match_id,competition_id,year,servingPlayer as player_id,
    sum(Case is1stServe when 1 then 1 else 0 end) As 1stServeAce,
    sum(Case is1stServe when 0 then 1 else 0 end) As 2ndServeAce
FROM point_by_point
WHERE isAce = 1
GROUP BY match_id,competition_id,year,servingPlayer
```

- **Numero di break point giocati**

```
SELECT match_id, competition_id, year, playerA_id AS player_id,
    COUNT(*) AS BreakPlayed
FROM point_by_point
WHERE isBreakPoint = 1
GROUP BY match_id, competition_id, year, playerA_id
UNION
SELECT match_id, competition_id, year, playerB_id AS player_id,
    COUNT(*) AS BreakPlayed
FROM point_by_point
WHERE isBreakPoint = 1
GROUP BY match_id, competition_id, year, playerB_id
```

- **Numero di punti giocati**

```
SELECT match_id, competition_id, year, playerB_id AS player_id,
    COUNT(1) AS PointPlayed
FROM point_by_point
GROUP BY match_id, competition_id, year, playerB_id
UNION
SELECT match_id, competition_id, year, playerA_id AS player_id,
    COUNT(1) AS PointPlayed
FROM point_by_point
GROUP BY match_id, competition_id, year, playerA_id
```

- **Numero di break point vinti**

```
SELECT competition_id, match_id, year, scoredBy AS player_id,
    COUNT(*) AS BreakWin
FROM point_by_point
WHERE isBreakPoint = 1 AND scoredBy != servingPlayer
GROUP BY competition_id, match_id, year, scoredBy
```

- **Numero di doppi falli**

```
SELECT match_id, competition_id, year, servingPlayer as player_id,
    COUNT(*) As doubleFaultCount
FROM point_by_point
WHERE errorType_A = 5 OR errorType_B = 5
GROUP BY match_id, competition_id, year, servingPlayer
```

- **Numero di punti al servizio**

```
SELECT match_id, competition_id, year, servingPlayer as player_id,
    sum(Case is1stServe when 1 then 1 else 0 end) As 1stServePoint,
    sum(Case is1stServe when 0 then 1 else 0 end) As 2ndServePoint
FROM point_by_point
WHERE servingPlayer = scoredBy AND isAce = 0 AND
    (winType_A != 3 AND winType_B !=3)
GROUP BY match_id, competition_id, year, servingPlayer
```

- **Numero di errori a rete**

```
SELECT match_id, competition_id, year, playerA_id AS player_id,
    COUNT(*) AS ErrorNetPoint
FROM point_by_point
WHERE position_A = 2 AND (errorType_A = 3 OR errorType_A = 4)
GROUP BY match_id, year, playerA_id
UNION
SELECT match_id, competition_id, year, playerB_id AS player_id,
    COUNT(*) AS ErrorNetPoint
FROM point_by_point
WHERE position_B = 2 AND (errorType_B = 3 OR errorType_B = 4)
GROUP BY match_id, year, playerB_id
```

- **Numero di ace subiti**

```
SELECT match_id, competition_id, year, playerB_id AS player_id,
    COUNT(1) AS AceAgainst
FROM point_by_point
WHERE servingPlayer = playerA_id AND isAce=1 AND
( errorType_A  != 5 )
GROUP BY match_id, competition_id, year, servingPlayer
UNION
SELECT match_id, competition_id, year, playerA_id AS player_id,
    COUNT(1) AS AceAgainst
FROM point_by_point
WHERE servingPlayer = playerB_id AND isAce =1 AND
( errorType_B != 5 )
GROUP BY match_id, competition_id, year, servingPlayer
```

- **Numero di potenziali ritorni**

```
SELECT match_id, competition_id, year, playerB_id AS player_id,
    COUNT(1) AS PotentialReturnPlayed
FROM point_by_point
WHERE servingPlayer = playerA_id AND ( errorType_A != 5 )
GROUP BY match_id, competition_id, year, servingPlayer
UNION
SELECT match_id, competition_id, year, playerA_id AS player_id,
    COUNT(1) AS PotentialReturnPlayed
FROM point_by_point
WHERE servingPlayer = playerB_id AND ( errorType_B != 5 )
GROUP BY match_id, competition_id, year, servingPlayer
```

- **Numero di break point salvati**

```
SELECT competition_id, match_id, year, scoredBy AS player_id,
    COUNT(*) AS BreakSaved
FROM point_by_point
WHERE isBreakPoint = 1 AND scoredBy = servingPlayer
GROUP BY competition_id, match_id, year, scoredBy
```

- **Numero di ritorni effettuati**

```
SELECT match_id, competition_id, year, playerB_id AS ‘player_id‘,
 COUNT(1) AS ‘ReturnPlayed‘,
 sum(case isFirstServe when 1 then 1 else 0 end) AS firstReturn,
 sum(case isFirstServe when 0 then 1 else 0 end) AS secondReturn
FROM point_by_point
WHERE servingPlayer = playerA_id and isAce = 0
AND errorType_A <> 5 AND winType_A <> 3
GROUP BY match_id,competition_id,year,servingPlayer
UNION
SELECT match_id, competition_id, year, playerA_id AS ‘player_id‘,
 COUNT(1) AS ReturnPlayed,
 sum(case isFirstServe when 1 then 1 else 0 end) AS firstReturn,
 sum(case isFirstServe when 0 then 1 else 0 end) AS secondReturn
FROM point_by_point
WHERE servingPlayer = playerB_id and isAce = 0
AND errorType_B <> 5 AND winType_B <> 3
GROUP BY match_id, competition_id, year, servingPlayer
```

- **Numero di servizi**

```
SELECT match_id, competition_id, year, servingPlayer as player_id,
 COUNT(*) as serveCount,
 sum(Case is1stServe when 1 then 1 else 0 end ) As 1stServeCount,
 sum(Case is1stServe when 0 then 1 else 0 end) As 2ndServeCount
FROM point_by_point
GROUP BY match_id, competition_id, year, servingPlayer
```

- **Numero di servizi non ritornati (punti diretti)**

```
SELECT  match_id, competition_id, year, servingPlayer as player_id,
 sum(Case is1stServe when 1 then 1 else 0 end) As 1stServeUnreturn,
 sum(Case is1stServe when 0 then 1 else 0 end) As 2ndServeUnreturn
FROM point_by_point
WHERE servingPlayer = scoredBy AND isAce = 0 AND
(winType_A = 3 or winType_B = 3)
GROUP BY match_id, competition_id, year, servingPlayer
```

- **Numero di punti vinti a rete**

```
SELECT match_id, competition_id, year, playerA_id AS player_id,
    COUNT(*) AS TotalNetPoint,
SUM(CASE scoredBy WHEN playerA_id THEN 1 ELSE 0 END) NetPointWon
FROM point_by_point
WHERE position_A = 2
GROUP BY match_id, year, playerA_id
UNION
SELECT match_id, competition_id, year, playerB_id AS player_id,
    COUNT(*) AS TotalNetPoint,
SUM(CASE scoredBy WHEN playerB_id THEN 1 ELSE 0 END) NetPointWon
FROM point_by_point
WHERE position_B = 2
GROUP BY match_id, year, playerB_id
```

- **Numero di non-ritorni**

```
SELECT match_id, competition_id, year, playerB_id AS player_id,
    COUNT(1) AS UnReturnPlayed,
    sum(case is1stServe when 1 then 1 else 0 end) AS 1stUnReturn,
    sum(case is1stServe when 0 then 1 else 0 end) AS 2ndUnReturn
FROM point_by_point
WHERE servingPlayer = playerA_id AND isAce = 0 AND
    ( errorType_A != 5 ) AND ( winType_A = 3 )
GROUP BY match_id, competition_id,year, servingPlayer
UNION
SELECT match_id,competition_id,year,playerA_id AS player_id,
    COUNT(1) AS UnReturnPlayed,
    sum(case is1stServe when 1 then 1 else 0 end) AS 1stUnReturn,
    sum(case is1stServe when 0 then 1 else 0 end) AS 2ndUnReturn
FROM point_by_point
WHERE servingPlayer = playerB_id AND isAce = 0 AND
    ( errorType_B != 5 ) AND ( winType_B = 3 )
GROUP BY match_id, competition_id, year, servingPlayer
```

5.6 Creazione del dato aggregato e dei nuovi Indici/KPI

Dopo aver creato le viste potevo quindi lavorare su delle tabelle che avevano i dati grezzi, si è reso necessario allora aggregare i dati.
Per prima cosa ho dovuto "raggruppare" per codice giocatore, identificativo del torneo e anno, così da avere tutti i possibili raggruppamenti e le somme dei dati, per ogni raggruppamento.
per questo studio abbiamo ideato diverse metriche, e quattro indici per poter confrontare e quantificare la forza di un giocatore nei principali fondamentali del tennis.
Le metriche che ho creato ed utilizzato sono:

- Percentuale di ace
  - %Ace1st = numero di ace sul primo servizio / numero di primi servizi giocati.
  - %Ace2nd = numero di ace sui secondi servizi / numero di secondi servizi.
- Percentuale di punti fatti
  - %PntOn1st = numero di punti fatti dopo un primo servizio / numero di primi servizi.
  - %PntOn2nd = numero di punti fatti dopo un secondo servizio / numero di secondi servizi.
- Percentuale di servizi non ritornati
  - %UnreturnOn1st = numero di servizi non ritornati / numero di primi servizi.
  - %UnreturnOn2nd = numero di servizi non ritornati / numero di secondi servizi.
- Percentuale di Doppi Falli
  - %DF = numero di doppi falli commessi/ numero di servizi

- Percentuali di ritorni effettuati
  - %return = numero di ritorni riusciti / numero di ritorni potenziali.
  - %returnOn1st = numero di ritorni riusciti contro una prima al servizio / numero di primi ritorni potenziali.
  - %returnOn2nd = numero di ritorni riusciti contro una seconda al servizio / numero di secondi ritorni potenziali.
- Percentuali di ace subiti
  - %aceagainst = numero di ace subiti / numero di potenziali servizi contro.
- Percentuali di ritorni non riusciti
  - %unretun = numero di servizi a cui il giocatore non è riuscito a rispondere / numero di potenziali servizi contro.
  - %unretunOn1st = numero di primi servizi a cui il giocatore non è riuscito a rispondere / numero di potenziali primi servizi contro.
  - %unretunOn1st = numero di secondi servizi a cui il giocatore non è riuscito a rispondere / numero di potenziali secondi servizi contro.
- Percentuale di approcci a rete
  - %netApproach = numero di volte in cui il giocatore è andato a rete / numero di punti totali giocati
- Percentuale di errori a rete
  - %errorOnNet = numero di errori commessi a rete / numero di volte in è andato a rete
- Percentuale di punti vinti a rete
  - %pointOnNet = numero di punti vinti andando a rete / numero di volte in cui è andato a rete
- Percentuali di break point vinti
  - %breakpointwon = numero di break point vinti (il giocatore non stava servendo) / numero di break point giocati

- Percentuali di break point persi
  - %breakpointlose = numero di break point persi (il giocatore non stava servendo) / numero di break point giocati
- Percentuali di break point salvati
  - %breakpointsaved = numero di break point vinti (il giocatore stava servendo) / numero di break point giocati

Una volta ottenute queste metriche saremmo già in grado di fare dei confronti e abbozzare delle statistiche, ma nel nostro caso abbiamo voluto ancora aggregare questi dati creando degli indici, andando a coprire i quattro fondamentali ovvero, il servizio, la risposta, il break point e il gioco a rete.
Per effettuare il passaggio da metrica a indice ho trasformato le variabili calcolate prima andando a moltiplicare le frequenze dei singoli giocatori per il valore medio generale che chiameremo "peso".
Questo peso viene calcolato sommando tutti i valori di quella metrica dividendolo per la somma del dato generale, andando a vedere il grado di indidenza di quel tipo di dato.
Ad esempio, nel caso dei doppi falli:
indexDF = frequenzaDF * valoremedioDF
valoremedioDF = somma(doppi falli)/somma(servizi giocati) = 0.09

Nome	%DF	indexDF
Player1	3,15	3,15*0,09
Player2	2,98	2,98*0,09

Tabella 5.1: Tabella di esempio in cui viene creato un indice.

Dopodichè avendo i singoli indici per ogni variabile, gli aggregati ulteriormente andando a creare così un indice unico che rappresenti la forza nel fondamentale.
Il dato aggregato finale viene costruito andando a sommare gli indici positivi (come ace, punti fatti, servizi non ritornati) e sottraendo quelli negativi (come i doppi falli), andando a creare un dato pesato sulla base della frequenza con il dato si verifica(es. %1st e %2nd).

5.6.1 Serve Index

Il serve index rappresenta un calcolo oggettivo per misurare la forza di un giocatore al servizio.
Per il calcolo di questo indice si utilizzano alcune delle metriche descritte prima, in questo caso si valutano l'indice con cui un giocatore esegue un ace, l'indice di efficacia a fare punti quando un giocatore è al servizio, l'indice con cui un giocatore consegue punti diretti servendo e l'indice che rappresenta gli errori.

SERVE_INDEX = %1st * (ACE1st + PNT1st + UNRETURN1st) + %2nd * (ACE2nd + PNT2nd + UNRETURN2nd) - INDEXDF

Nome	%1st	%2nd	ACE1st	ACE2nd	PNT1st	PNT2nd
Player1	0,73	0,27	0,38	0	16,36	15,29
Player2	0,63	0,37	0,62	0,02	18,64	16,06
Player3	0,60	0,40	2,40	0,03	13,75	15,50

Tabella 5.2: Tabella di esempio in cui viene creato il serve index.

Nome	UNRETURN1st	UNRETURN2nd	DOUBLE_FAULT
Player1	6,96	2,77	2,45
Player2	4,96	1,61	2,43
Player3	1,93	1,80	2,44

Tabella 5.3: Continuazione della tabella di esempio in cui viene creato il serve index.

Nome	SERVE INDEX
Player1	= 0,73*(0,38 + 16,36 + 6,96) + 0,27*(15,29 + 2,77) - 2,45
Player2	= 0,63*(0,37 + 18,64 + 4,96) + 0,37*(0,02 + 16,06 + 1,61) - 2,43
Player3	= 0,60*(0,40 + 13,75 + 1,93) + 0,40*(0,03 + 15,50 + 1,80) - 2,44

Tabella 5.4: Continuazione della tabella di esempio in cui viene creato il serve index.

5.6.2 Return Index

Il return index rappresenta, in modo oggettivo, la forza di un giocatore nel rispondere al servizio.
Per calcolare l'indice ho utilizzato diverse metriche descitte prima, utilizzando, nello specifico per questo tipo, l'indice che valuta la forza nella risposta ad un servizio, la debolezza nel subire punti diretti contro un servizi e la debolezza nel subire gli ace.
In particolare ho utilizzato questa formula:

RETURN_INDEX =%RETURN * (RETURN_1ST + RETURN_2ND) - %UNRETURN * (UNRETURN_1ST + UNRETURN_2ND) - ACE_AGAINST

Nome	RETURN	RETURN1st	RETURN2nd	ACE_AGAINST
Player1	51,19	39,50	13,80	0,70
Player2	51,45	41,74	12,48	0,26
Player3	48,95	45,70	10,15	0,32

Tabella 5.5: Tabella di esempio in cui viene creato il Return index.

Nome	UNRETURN	UNRETURN1st	UNRETURN2nd
Player1	3,91	57,44	5,80
Player2	5,13	56,52	6,06
Player3	5,61	63,11	4,20

Tabella 5.6: Continuazione della tabella di esempio in cui viene creato il return index.

Nome	RETURN INDEX
Player1	= 51,19*(39,50 + 13,80) - 3,91*(57,44 + 5,80) - 0,70
Player2	= 51,45*(41,74 + 12,48) - 5,13*(56,26 + 6,06) - 0,26
Player3	= 48,95*(45,70 + 10,15) - 5,61*(63,11 + 4,20) - 0,32

Tabella 5.7: Continuazione della tabella di esempio in cui viene creato il return index.

5.6.3 Volley Index

Il volley index è un modo di rappresentare in maniera oggettiva la forza di un giocatore nel giocare a rete.
In particolare per questo specifico indice ho aggregato insieme diversi indici, ovvero quello che descrive la forza nel vincere i punti quando un giocatore va a rete e quello che indica la debolezza di un giocatore a commettere errori sempre sotto rete.
Tutti questi indici sono stati poi pesati per la frequenza con cui il giocatore si approccia a rete, poiché un giocatore potrebbbe andare a rete molto frequentemente e fare pochi punti, questo lo rende più debole rispetto ad un altro che invece si approccia meno, ma che vince più punti.
In particolare ho utilizzato la seguente formula matematica:

VOLLEY_INDEX = NET_APPROACH * POINT_ON_NET – ERROR_ON_NET * NET_APPROACH

Nella seguente tabella, abbiamo un esempio di dati utilizzati per calcolare l'indice:

Nome	NET_APPROACH	POINT_ON_NET	ERROR_ON_NET
Player1	0,86	45,07	0,23
Player2	2,07	46,88	0,38
Player3	3,05	40,68	0,56

Tabella 5.8: Tabella di esempio in cui viene creato il volley index.

Partendo quindi dai dati presenti nella tabella 5.8, ho quindi calcolato i valori come si può vedere nella tabella 5.9.

Nome	VOLLEY_INDEX
Player1	= 0,86 * 45,07 - 0,86 * 0,23
Player2	= 2,07 * 46,88 - 2,07 * 0,38
Player3	= 3,05 * 40,68 - 3,05 * 0,56

Tabella 5.9: Continuazione tabella di esempio in cui viene creato il volley index.

5.6.4 Break Index

Il break index rappresenta la forza di un giocatore nel giocare i break point. Un giocatore può trovare a giocarsi un break point in due situazioni, una in vase di vantaggio, ovvero quando non stà servendo e si trova ad un punto dal vincere il game, l'altre è una situzuoine di svantaggio, ovvero quando il giocatore sta servendo e al suo avversario manca un punto per vincere il game, per cui deve "salvare i punti".
In questo indice ho quindi aggregato insisme gli indici che rappresentano la capacità di un giocatore di vincere i break point, la capacità di salvare i break point e la debolezza nel perderli.
Di seguito la formula che ho applicato per ottenere l'indice:

BREAK_INDEX = %POTENTIAL_BREAK * (BREAK_WIN - BREAK_LOSE) + BREAK_SAVED * (1 - %POTENTIAL_BREAK)

Nella seguente tabella, abbiamo un esempio di dati utilizzati per calcolare l'indice:

Nome	%POTENTIAL BREAK	BREAK WIN	BREAK LOSE	BREAK SAVED
Player1	0,47	11,44	41,25	17,19
Player2	0,85	6,06	56,78	4,9
Player3	0,5	10,3	44,55	16,33

Tabella 5.10: Tabella di esempio in cui viene creato il Break index.

Partendo quindi dai dati presenti nella tabella 5.10, ho quindi calcolato i valori come si può vedere nella tabella 5.11.

Nome	BREAK INDEX
Player1	= 0,47*(11,44 - 41,25) + 17,19*(1 - 0,47)
Player2	= 0,85*(6,06 - 56,78) + 4,9*(1 - 0,85)
Player3	= 0,5*(10,33 - 44,55) + 16,33*(1 - 0,5)

Tabella 5.11: Continuazione tabella di esempio in cui viene creato il Break index.

5.6.5 Overall Index

L'overall index è una rappresentazione generalizzata della forza di un giocatore, andando a considerare la forza nei singoli fondamentali pesando la percentuale in cui la singola situazione si verifica.

OVERALL_INDEX = %SERVE*SERVE_INDEX + %RETURN*RETURN_INDEX + %VOLLEY*VOLLEY_INDEX + %BREAK*BREAK_INDEX

Nelle seguenti tabelle, abbiamo un esempio di dati utilizzati ed esempio di calcoli effettuati:

Nome	%SERVE	%RETURN	%VOLLEY	%BREAK
Player1	0,51	0,48	0,07	0,05
Player2	0,47	0,50	0,17	0,07
Player3	0,54	0,45	0,25	0,03

Tabella 5.12: Tabella di esempio in cui viene creato l'overall index.

Nome	SERVE INDEX	RETURN INDEX	VOLLEY INDEX	BREAK INDEX
Player1	19,73	2480,67	38,74	94,93
Player2	19,35	2468,41	96,19	57,62
Player3	17,14	2356,01	122,39	91,04

Tabella 5.13: Continuazione della tabella di esempio in cui viene creato l'overall index.

Nome	OVERALL INDEX
Player1	= 0,51 * 19,73 + 0,48 * 2480,67 + 0,07 * 38,74 + 0,05 * 94,93
Player2	= 0,47 * 19,35 + 0,50 * 2468,41 + 0,17 * 96,19 + 0,07 * 57,62
Player3	= 0,54 * 17,14 + 0,45 * 2356,01 + 0,25 * 122,39 + 0,03 * 91,04

Tabella 5.14: Continuazione della tabella di esempio in cui viene creato l'overall index.

Di seguito possiamo vedere degli estratti di codice che rappresentano tutti i passaggi che abbiamo espresso nei paragrafi precedenti, e che a partire dalle interrogazioni alle viste portano al calcolo del Serve Index fatto per ogni giocatore, per poter fare un confronto tra i giocatori.

Inizialmente si fanno le query sul databse per ottenere i dati, quindi definite le viste che si vogliano interrogare e la query SQL si lancia il comando "read sql" di python e ottenere per ogni vista un dataframe che poi verranno concatenati, così da ottenerne uno unico.
Inoltre per ogni colonna riempio con il valore 0 dove il dato è mancante.

```
ESEMPIO DI CREAZIONE SERVE INDEX
serveIndexView = ['vw_serve','vw_ace', 'vw_point_on_service',
                                                    'vw_unreturnserve','vw_doublefault' ]
for view in serveIndexView:
    sql_select_Query = "select * from %s" %(view)
    dataframe = pd.read_sql(sql_select_Query, conn)

serveIndexDataframe = dataframe.groupby('player_id').sum()
for idx in serveIndexDataframe.keys():
    serveIndexDataframe[idx] = serveIndexDataframe[idx].fillna(0)
```

In questa sezione viene calcolato, per ogni tipo di dato, il "peso", ovvero la frequenza media con cui il dato si verifica, andando a sommare tutta colonna della relativa tipologia(es Ace) dividendola per la somma del dato generale(es Servizi giocati).
Per effettuare la somma ho utilizato il metodo sum() dei dataframe, mentre i pesi vengono salvati in una struttura indicizzata utile per ottenere facilmente il dato.

```
weights['ace1'] = serveIndexDataframe['firstServeAce'].sum()/
    serveIndexDataframe['firstServeCount'].sum()

weights['ace2'] = serveIndexDataframe['secondServeAce'].sum()/
    serveIndexDataframe['secondServeCount'].sum()

weights['pntwon1'] = serveIndexDataframe['firstServePoint'].sum()/
    serveIndexDataframe['firstServeCount'].sum()

weights['pntwon2'] = serveIndexDataframe['secondServePoint'].sum()/
    serveIndexDataframe['secondServeCount'].sum()

weights['unreturn1'] = serveIndexDataframe['firstServeUnreturn'].sum()/
    serveIndexDataframe['firstServeCount'].sum()

weights['unreturn2'] = serveIndexDataframe['secondServeUnreturn'].sum()/
    serveIndexDataframe['secondServeCount'].sum()

weights['df'] = serveIndexDataframe['doubleFaultCount'].sum()/
    serveIndexDataframe['serveCount'].sum()
```

In questa sezione di codice vengono create le metriche vere e proprie, per ciascuna di esse, si effettuano le operazioni matematiche e si salvano in una nuova colonna sempre dello stesso dataframe.
Queste operazioni vengono effettuate per ogni riga del dataframe, utilizzando il metodo "apply", che applica la funzione passata come parametro ad ognuna di esse.
Come funzione passata ad apply, ho utilizzato una funzione locale utilizzando il metodo labda, il quale crea una funzione, che in questo caso compie i calcoli.

```
serveIndexDataframe['%ace1'] = serveIndexDataframe.apply(
    lambda row: 100*row['firstServeAce']/row['firstServeCount'], axis=1 )

serveIndexDataframe['%ace2'] = serveIndexDataframe.apply(
    lambda row: 100*row['secondServeAce']/row['secondServeCount'], axis=1 )

serveIndexDataframe['%pntwon1'] = serveIndexDataframe.apply(
    lambda row: 100*row['firstServePoint']/row['firstServeCount'], axis=1 )

serveIndexDataframe['%pntwon2'] = serveIndexDataframe.apply(
    lambda row: 100*row['secondServePoint']/row['secondServeCount'], axis=1 )

serveIndexDataframe['%unreturn1'] = serveIndexDataframe.apply(
    lambda row: 100*row['firstServeUnreturn']/row['firstServeCount'], axis=1 )

serveIndexDataframe['%unreturn2'] = serveIndexDataframe.apply(
    lambda row: 100*row['secondServeUnreturn']/row['secondServeCount'], axis=1 )

serveIndexDataframe['%df'] = serveIndexDataframe.apply(
    lambda row: 100-100*row['doubleFaultCount']/row['serveCount'], axis=1 )
```

Dopo aver calcolato le metriche, il passo successivo e ottenere gli indici, per calcolarli, come abbiamo detto nel relativo paragrafo, bisogna moltiplicare la metriche per il relativo "peso" calcolato prima. Questa operazione va ripetuta per ogni riga e il risultato viene salvato in una nuova colonna appena al dataframe originale.

```
serveIndexDataframe['index_ace1'] = serveIndexDataframe.apply(
                        lambda row: row['%ace1']*weights['ace1'], axis=1 )

serveIndexDataframe['index_ace2'] = serveIndexDataframe.apply(
                        lambda row: row['%ace2']*weights['ace2'], axis=1 )

serveIndexDataframe['index_pntwon1'] = serveIndexDataframe.apply(
                        lambda row: row['%pntwon1']*weights['pntwon1'], axis=1 )

serveIndexDataframe['index_pntwon2'] = serveIndexDataframe.apply(
                        lambda row: row['%pntwon2']*weights['pntwon2'], axis=1 )

serveIndexDataframe['index_unreturn1'] = serveIndexDataframe.apply(
                        lambda row: row['%unreturn1']*weights['unreturn1'], axis=1 )

serveIndexDataframe['index_unreturn2'] = serveIndexDataframe.apply(
                        lambda row: row['%unreturn2']*weights['unreturn2'], axis=1 )

serveIndexDataframe['index_df'] = serveIndexDataframe.apply(
                        lambda row: row['%df']*weights['df'], axis=1 )
```

Nella sezione che segue ho calcolato allo stesso modo delle metriche le percentuali di primi e secondi servizi., ho quindi utilizzato nuovamente apply, con dentro una lambda function.

```
serveIndexDataframe['%1st'] = serveIndexDataframe.apply(
                        lambda row: row['firstServeCount']/row['serveCount'], axis=1 )

serveIndexDataframe['%2nd'] = serveIndexDataframe.apply(
                        lambda row: row['secondServeCount']/row['serveCount'], axis=1 )
```

Successivamente ho applicato sempre utilizzando apply una lambda function in cui ho inserito la formula matematica del serve index.

```
serveIndexDataframe['serveIndex'] = serveIndexDataframe.apply(
    lambda row:
        float(row['%1st'])*(
        float(row['index ace1'])
        + float(row['index unreturn1'])
        + float(row['index pntwon1']))
        + float(row['%2nd'])*(
        float(row['index ace2'])
        + float(row['index unreturn2'])
        + float(row['index pntwon2']))
        - float(row['index df']),
    axis=1 )
```

Infine in questo blocco di codice ho applicato una normalizzazione dei valori affinchè possano essere comparabili in una scala percentuale.
Per calcolare il dato normalizzato ho quindi diviso il dato per il valore massimo che assume tra gli altri della colonna, utilizzado il metodo max() dei dataframe, e poi moltiplicando per cento.

```
serveIndexDataframe['norm index ace1'] = serveIndexDataframe.apply(
    lambda row: round( 100 * row['index ace1'] /
                                serveIndexDataframe['index ace1'].max(),2), axis=1 )

serveIndexDataframe['norm index ace2'] = serveIndexDataframe.apply(
    lambda row: round( 100 * row['index ace2'] /
                                serveIndexDataframe['index ace2'].max(),2), axis=1 )

serveIndexDataframe['norm index pntwon1'] = serveIndexDataframe.apply(
    lambda row: round( 100 * row['index pntwon1'] /
                                serveIndexDataframe['index pntwon1'].max(),2), axis=1 )

serveIndexDataframe['norm index pntwon2'] = serveIndexDataframe.apply(
    lambda row: round( 100 * row['index pntwon2'] /
                                serveIndexDataframe['index pntwon2'].max(),2), axis=1 )

serveIndexDataframe['norm index unreturn1'] = serveIndexDataframe.apply(
    lambda row: round( 100 * row['index unreturn1'] /
                                serveIndexDataframe['index unreturn1'].max(),2), axis=1 )

serveIndexDataframe['norm index unreturn2'] = serveIndexDataframe.apply(
    lambda row: round(100*row['index unreturn2'] /
                                serveIndexDataframe['index unreturn2'].max(),2), axis=1 )

serveIndexDataframe['norm index df'] = serveIndexDataframe.apply(
    lambda row: round(100*row['index df'] /
                                serveIndexDataframe['index df'].max(),2), axis=1 )
```

5.7 Creazione delle grafiche comparative

Dopo aver creato gli indici analiticamente non rimane che creare delle grafiche per mostrare i risultati ed avere un primo riscontro visivo.

Per la creazione delle grafiche ho utilizzato la libreria graphviz di python che permette la creazione di diverse tipologie di grafici. Nel mio caso ho utilizzato *radar chart* per quegli indici che possiedono almeno 4 attributi, mentre per quelli che ne hanno meno ho creato degli *istogrammi a barre.*

L'idea dietro a queste grafiche è che si possa fare un rapido confronto sui punti di forza e di debolezza di un giocatore, questo può essere fatto facendo un confronto diretto verso un altro giocatore, oppure facendo un confronto tra il giocatore selezionato e il giocatore medio.

Il giocatore medio è il prototipo di giocatore che viene creato sulla base dei valori medi dei giocatori andando a calcolare il valore medio per ogni dato "grezzo" e poi calcolando il relativo indice.
In basso due esempi di creazione di grafici, sia "bar plot" che "radar chart".

```
ESEMPIO RADAR CHART
chartdf = chartDataFrame.rename(columns={
    'norm index ace1': 'Aces on 1st Serve',
    'norm index ace2': 'Aces on 2nd Serve',
    'norm index pntwon1': 'Points on 1st Serve',
    'norm index pntwon2': 'Points on 2nd Serve',
    'norm index unreturn1': 'Unreturned 1st Serve',
    'norm index unreturn2': 'Unreturned 2nd Serve',
    'norm index df': 'Double Fault Strenght'
})

selectedPlayerList = []
for row in chartdf.to_numpy():
    tuple = (row[0],row[1],row[2],row[3],row[4],row[5],
    row[6],row[0])
    selectedPlayerList.append(tuple)

label_loc = np.linspace(start=0, stop= 2 * np.pi, num=len(categories))

plt.figure(figsize=(8, 8))
plt.subplot(polar=True)

listindex = chartdf.index

for row,idx in zip(selectedPlayerList,listindex):
    plt.plot( label_loc, row, label=idx )

plt.title('Serve Index', size=20)
lines, labels = plt.thetagrids(np.degrees(label_loc),
                               labels=categories)
plt.legend()
plt.show()
```

```
ESEMPIO BAR PLOT

plotdata = pd.concat(
    [
        chartDataFrame['Net Approach'],
        chartDataFrame['Point Won on Net'],
        chartDataFrame['Error']
    ],
axis=1)

Plotdata.plot(kind="bar",figsize=(15, 8))
plt.title("Volley Index")
plt.xlabel("Player")
plt.ylabel("Scores")
plt.show()
```

5.7.1 Serve chart

Il serve chart è un radar chart che rappresenta il serve index, ovvero quanto un giocatore sia forte nel fondamentale del servizio, in particolare troviamo le diverse metriche che lo compongono, come:

- Points on 1st Serve
- Points on 2ns Serve
- Unreturned 1st Serve
- Unreturned 2nd Serve
- Double fault Strenght
- Aces on 1st Sreve
- Aces on 2nd Serve

Nella figura 5.4 possiamo vedere un esempio di radar chart dove si mettono a confronto due giocatori andando a confrontare la loro forza nei vari punti sopracitati.
Inoltre se fosse selezionato un solo giocatore si farebbe un confronto, allo stesso modo di un altro giocatore, con il "giocatore medio", andando a vedere se ha statistiche sopra o sotto la media.
In questo caso possiamo vedere come il giocatore contrassegnato dalla linea blu sia più forte rispetto al giocatore "arancione" in alcuni valori, mentre risulta più debole in altri, ad esempio, è forte nel vincere i punti dopo un primo servizio e nel fare punti diretti sia sul primo che secondo servizio, mentre è più debole nel vincere i punti dopo una seconda di servizio ed è inoltre più debole sui doppi falli, ovvero ne commette di più.

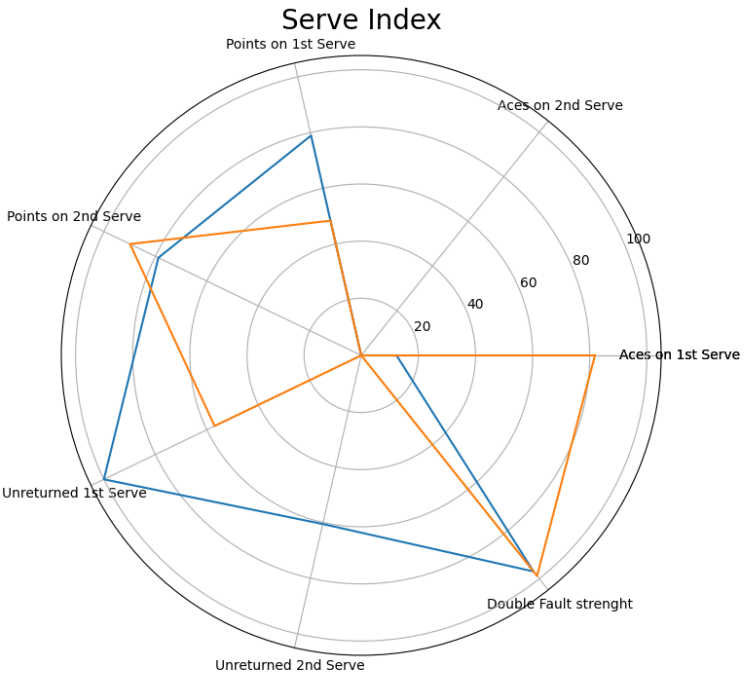

Figura 5.4: grafica relativa al Serve index

5.7.2 Return chart

Il Return chart è un radar chart che rappresenta il return index, ovvero quanto un giocatore sia forte nel fondamentale della risposta al servizio, in particolare troviamo le diverse voci che lo compongono, come:

- Returns on 1st Serve
- Returns on 2nd Serve
- Unreturned 1st Serve
- Unreturned 2nd Serve
- Strenght against Unreturn
- Strenght against Unreturn Aces
- Return Played

Nella figura 5.5 possiamo vedere un esempio di radar chart dove si mettono a confronto due giocatori andando a confrontare la loro forza nei vari punti sopracitati.
Inoltre se fosse selezionato un solo giocatore si farebbe un confronto, allo stesso modo di un altro giocatore, con il "giocatore medio", andando a vedere se ha statistiche sopra o sotto la media.
In questo caso, ad esempio, possiamo vedere come in generale il giocatore contrassegnato dalla linea blu sia più forte nel rispondere ai servizi rispetto al giocatore "arancione", in particolare possiamo vedere che ha valori più alti in risposta ai secondi servizi rispetto all'arancione che sa rispondere meglio alle prime servizio; inoltre possiamo notare che l'arancione che subisce più punti diretti, in quanto ha un basso valore di "unreturned" soprattutto sulle prime servizi rispetto al blu, ed in più subisce anche più ace contro.

Figura 5.5: grafica relativa al Return index

5.7.3 Break chart

Il Break chart è un bar chart che rappresenta il break index, ovvero quanto un giocatore sia forte in situazioni chiave come i break-point, in particolare troviamo le diverse voci che lo compongono, come:

- Break Win
- Break Lose
- Break Saved

Nella figura 5.6 possiamo vedere un esempio di bar chart dove si mettono a confronto due giocatori andando a confrontare la loro forza nei vari punti sopracitati. In questo caso possiamo vedere come il giocatore di destra(federer) sia più debole rispetto al giocatore di sinistra poichè quest'ultimo ha vinto più palle break, inoltre ne ha perse molte meno ed infine possiamo vedere che il giocatore di sinistra ha "salvato" anche qualche palla break in una situazione di svantaggio, cosa che il giocatore di sinistra non è riuscito.

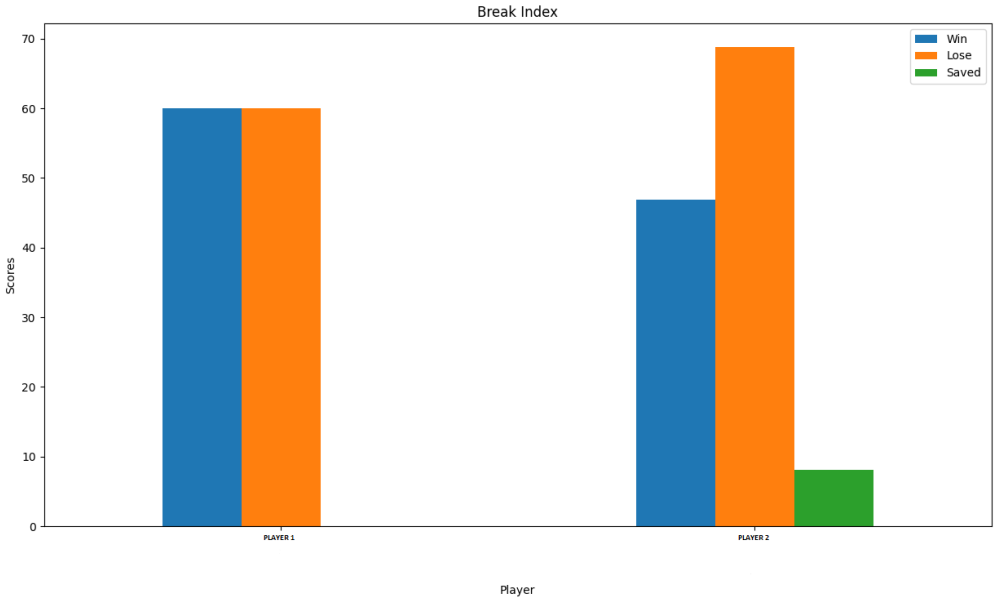

Figura 5.6: grafica relativa al Break index

5.7.4 Volley chart

Il Volley chart è un bar chart che rappresenta il volley index, ovvero quanto un giocatore sia forte nel giocare i punti a rete, in particolare troviamo le diverse voci che lo compongono, come:

- Net approach
- Points won on net
- Error

Nella figura 5.7 possiamo vedere un esempio di bar chart dove si mettono a confronto due giocatori andando a confrontare la loro forza nei vari punti sopracitati. In questo caso possiamo vedere come il giocatore a destra sia più propenso ad andare a rete rispetto al giocatore di sinistra, e questo fa si che sia più incline a commettere degli errori, e ad avere quindi qualche punto vinto in meno. Se dovessimo quindi dire chi sia il più forte potremmo dire che il giocatore di sinistra sia più bravo poichè nonostante vada poco a rete, quando capita è in grado di capitalizzare le poche chance che gli capitano.

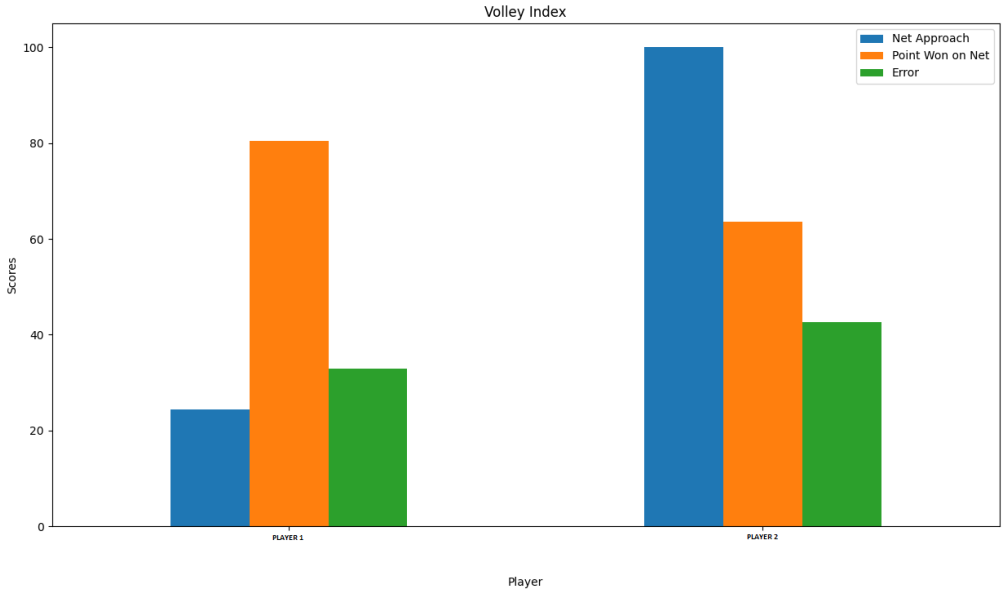

Figura 5.7: grafica relativa al Volley index

5.7.5 Overall chart

L'Overall chart è un radar chart che rappresenta l'overall index, ovvero quanto un giocatore sia forte in generale, in particolare troviamo le diverse voci che lo compongono, come:

- Volley Index
- Return Index
- Serve Index
- Break Index

Nella figura 5.8 possiamo vedere un esempio di radar chart dove si mettono a confronto due giocatori andando a confrontare la loro forza nei vari punti sopracitati. In questo caso possiamo vedere come il giocatore contrassegnato dalla linea blu sia più forte sul servizio, ma sia più debole rispetto al giocatore arancione negli altri indici come il break, volley e return.

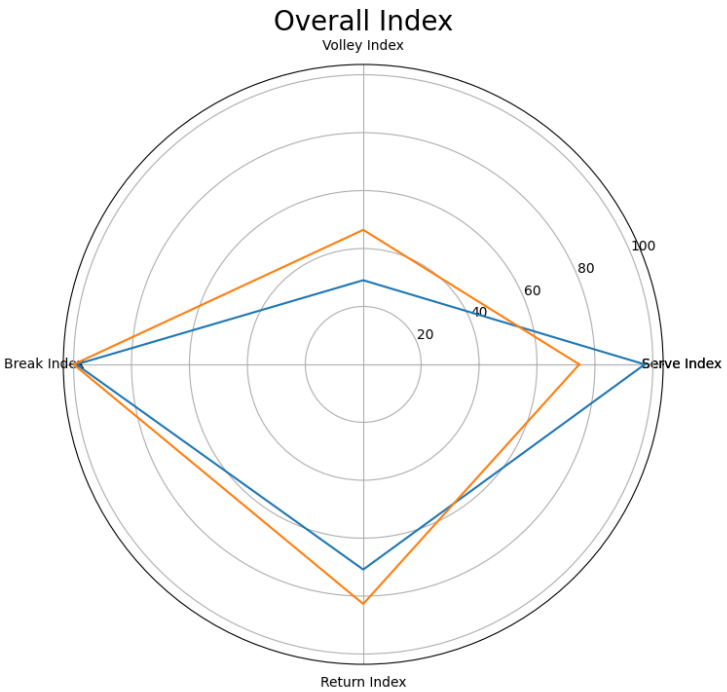

Figura 5.8: grafica relativa all'Overall index

5.8 Creazione interfaccia grafica e applicazione web

Il passo finale è stato quello di confezionare tutto il progetto e creare una applicazione semplice e intuitiva che possa dinamicamente generare e mostrare le grafiche di cui sopra.
Per fare questo ho utilizzato una libreria di python molto utile e potente, "Streamlit", che permette tramite codice di creare una applicazione web.
L'applicazione si presenta come mostrato nella figura 5.9:

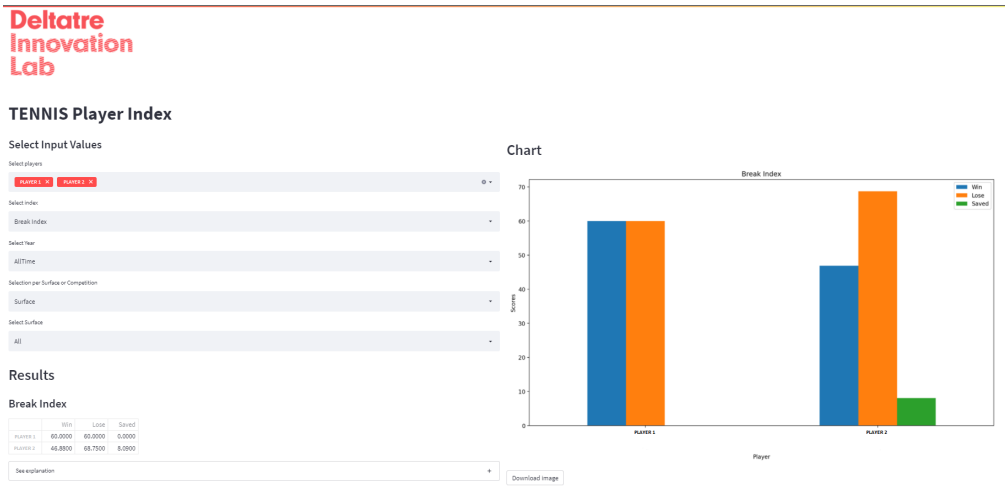

Figura 5.9: Screenshot della schermata principale dell'applicazione

Come possiamo vedere dalla figura 5.9, l'applicazione si compone di due colonne principali, la colonna destra, che mostra la grafica calcolata con il relativo tasto per poter scaricare l'immagine in formato jpg. La colonna di destra invece contiene dei menù a tendina e una text box, che permettono di impostare i parametri con cui poi verranno create le grafiche.
Nei menù di selezione è possibile applicare dei filtri per poter fare delle ricerche più "mirate", nello specifico è possibile scegliere se filtrare per competizione, o per superficie, filtrare per stagione o se visualizzare le statistiche "All Time" oppure filtrare per atleta.
L'applicazione è come abbiamo detto prima full-web e funziona facendo partire un mini "server" in locale sulla macchina utilizzando un comando di stramilit: *streamlit run C:/Users/matteo.gally/path/to/UserInterface.py*
Fatto questo si aprirà una finestra browser in ascolto sul localhost.

Di seguito un frammento di codice che descrive come è composta l'applicazione. Possiamo vedere come vengono definite le due colonne della pagina con "def col" e all'interno tutti i componenti descritti prima.
Nella colonna di sinistra (col1) possiamo vedere come siano presenti i menù a tendina(selectbox) e i text box(multiselect), che permettono di filtrare i dati, con cui viene ottenuto il dataframe con il metodo dataframe di Stramlit.

```
with col1:
    st.subheader('Select Input Values')
    #select player
    options = st.multiselect('Select players', player_table['lastName'])
    print('Players-> You selected: ', options)

    #select index
    listofindex = ['Overall Index','Break Index','Volley Index','Return Index','Serve Index']
    index = st.selectbox('Select index', listofindex)
    print('Index-> You selected: ', index)

    # select year
    listofindex = ['AllTime','2019', '2021']
    year = st.selectbox('Select Year', listofindex)
    print('Year-> You selected: ', year)

    # select surface/competition
    sur_comp = st.selectbox('Selection per Surface or Competition ', ['Surface','Competition'])

    if sur_comp == 'Surface':
        # select surface
        sur_index = ['All']
        for sur in surface.index:
            sur_index.append(sur)
        selection = st.selectbox('Select Surface', sur_index)
        selection = sur_comp + '-' + selection
        print('Surface-> You selected: ', selection)
    else:
        # select competition
        comp_index = ['All']
        for comp in competition.index:
            comp_index.append(comp)
        selection = st.selectbox('Select Competition', comp_index)
        selection = sur_comp + '-' + selection
        print('Competition-> You selected: ', selection)

    # obtain dataframe
    df = pd.DataFrame()
    desc = ''

    if index == 'Serve Index':
        df, chartdf, fig = Serve_Index.CreateServeIndex(options,year,selection)
        descr='Index that show streght and the weekness of a player in service fundamental.'
        values = 'We have some metrics:'
            '- Aces on 1st Serve: strenght to score aces on st service (higher better)'
            '- Aces on 2nd Serve: strenght to score aces on 2nd service (higher better)'
            '- Points on 1st serve: strength to score points on his 1st service (higher better)
            '- Points on 2nd serve: strength to score points on his 1st service (higher better)

    elif index == 'Break Index': ...

    elif index == 'Return Index': ...

    elif index == 'Volley Index': ...

    else: ...

    st.header('Results')
    st.subheader(index)
    st.dataframe(chartdf)
```

Nella colonna di destra(col2) invece viene semplicemente mostrato il grafico ottenuto richiamando la funzione corrispondente ai filtri, e in più con pulsante "download"(st.download_button) permete di scaricare l'immagine.

```
with col2:
    #print charts
    st.header('Chart')
    outpath = './%s.png' % (index)
    fig.savefig(outpath)

    st.pyplot(fig,fig.legend())

    #button for download charts
    with open(outpath, "rb") as file:
        btn = st.download_button(
            label = "Download image",
            data = file,
            file_name = outpath,
            mime = "image/png"
        )
```

Capitolo 6

Risultati e Casi d'uso

Come risultato finale ho ottenuto una applicazione web che mostra le grafiche calcolate in tempo reale, andando a modificare i menù a tendina e filtrando per nome i giocatori.

L'applicazione prende in input il path dove vengono salvati i file Json e li importa facendo partire uno script che elabora tutti i passaggi descritti nel capitolo "Svolgimento".
Una volta completato l'upload dei punti sul database e fatta partire l'applicazione si può ottenere immediatamente le statistiche e i grafici di cui abbiamo parlato in precedenza.
E' possibile interagire con l'applicazione andando a inserire i nomi dentro il "text box", e selezionare le statistiche relative ai giocatori in questione, inoltre nel caso in cui si selezioni un solo giocatore verrà calcolato il cosiddetto "giocatore medio", il quale mostra le statistiche medie dei giocatori, rendendo quindi possibile un confronto tra il giocatore selezionato con il resto dei giocatori, analizzando se abbia statistiche e performance sopra o sotto "la media".
Nel caso in cui invece si selezionino due o più giocatori verrà fatto un confronto diretto tra di essi mostrando le statistiche di ognuno.
Un altro tipo di filtro che è possibile applicare in aggiunta è quello relativo alle "stagioni", che permette di selezionare le statistiche dei giocatori "AllTime", ovvero riferito al loro storico completo, oppure circoscrivere la scelta a determinate annate. Inoltre applicando il filtro "Competizione o Superficie", dove una scelta esclude l'altra, si riesce a visualizzare le performance dei giocatori in determinati tornei o su determinate superfici.

Sul fondo della colonna di sinistra troviamo la tabella che esprime in formato numerico i dataframe calcolati, così da poter valutare oggettivamente la differenza di "performance" sulle metriche. Sotto è presente anche una info box che permette di dare informazioni riguardo all'indice che si sta mostrando, descrivendo i valori citati e come sono stati calcolati.

Nella colonna di destra invece vengono mostrati i grafici calcolati, con sotto un pulsante che da la possibilità di fare "Download" dell'immagine.

Questa applicazione è un esempio di come si comporta uno "Stats Engine" ovvero un applicativo che, dati in input dei parametri, permette di calcolare in tempo reale delle statistiche.
Questo tipo di strumento è molto potente e scalabile, infatti allo stesso modo in cui ho creato gli indici e le metriche sopracitate, se ne possono aggiungere molte altre a quelle già esistenti, che possono essere anch'esse calcolate in tempo reale.
Tutto questo semplicemente aggiungendo altre viste, per ottenere nuovi tipi di dato, e poi andando a studiare come calcolare, allo stesso modo degli altri, i nuovi indici.

L'applicazione è molto semplice da utilizzare e non ha bisogno di grandi competenze tecniche, infatti, come abbiamo detto prima l'applicativo funziona in completa autonomia, necessitando solo di essere impostato dall'utente. Nel caso in cui si tratti di configurazioni ex-novo, viene creato un database, il quale verrà generato in automatico lanciando un solo script, che eseguirà tutti i passaggi di processo dei dati.

Invece nel caso in cui l'utente abbia solo necessità di visualizzare delle statistiche l'applicativo si rivela ancora più semplice, poiché essendo una applicazione web non ha bisogno di installazione, ma solo di un portale web, quindi non richiede neanche risorse hardware di prim'ordine lato cliente, dato che il server che avvierà l'applicazione sarà a sua volta dislocato su una macchina esterna che manterrà attiva l'applicazione. Inoltre le uniche operazioni da fare sono esclusivamente di navigazione per cui molto semplici e intuitive, infatti basterà andare sul sito dell'applicazione e in automatico compariranno la schermate che abbiamo descritto in precedenza.

Essendo l'applicazione molto semplice da utilizzare si presta molto bene all'uso di tante professionalità sia tecniche che non:

- Potrebbe essere utilizzata da un professionista che ha bisogno di grafiche per poter fare delle analisi comparative tra giocatori da inserire all'interno di un blog specializzato o articoli di giornale sia generalisti che specializzati.
  Ad esempio nel caso in cui si voglia descrivere lo stato di forma di un giocatore, oppure presentare i due avversari di un match, o ancora indagare l'approccio ad un torneo di un giocatore verificando lo storico in quella competizione o il comportamento tenuto in altri tornei con la stessa superficie.

- Nell'ambito delle dirette testuali per esempio potrebbe essere utilizzata per creare delle grafiche da mostrare sui siti che raccontano le partite per creare una sorta di termometro della partita che permetta di descrivere l'andamento e il dominio di un giocatore sull'altro.

- Le emittenti televisive o le piattaforme streaming potrebbero utilizzare l'applicazione per mostrare grafiche e pillole statistiche durante un match, così da aiutare lo spettatore a capire il motivo per cui un giocatore sta vincendo, o come mai un altro si trovi in difficoltà.

- Allenatori o giocatori potrebbero utilizzare l'applicazione per capire i propri punti deboli e poter creare delle sessioni ad-hoc di allenamento per migliorarli, oppure potrebbe essere usata per studiare l'avversario che stanno per incontrare e andare a capire i punti forti e deboli per avere un vantaggio di gioco il giorno della partita.

- I trader che lavorano per i bookmakers potrebbero utilizzare l'applicazione per poter fare un confronto tra i giocatori e valutando anche lo storico delle loro statistiche essere aiutati a capire come potrebbe andare il match e quindi stabilire le quote.

- Ormai al giorno d'oggi si fa molta comunicazione sui social network, questa applicazione potrebbe essere utilizzata dai social media manager che gestiscono le pagine di giocatori o tornei per presentare dei match, oppure evidenziare alcune statistiche particolari di giocatori utilizzando le grafiche prodotte mettendole sui social.

Capitolo 7

Conclusioni e Lavori Futuri

In quanto progetto partito da zero, ed essendo il tempo avuto a disposizione limitato, il progetto si presta a molti miglioramenti, sia dal punto di vista del codice, in quanto potrebbe essere ripulito e ottimizzato per renderlo più performante e leggibile, sia per le funzionalità, che potrebbero aumentare.

Sicuramente un piccolo miglioramento, che è già stato discusso con l'azienda e che si verificherà a breve, è quello di aggiornare la versione di mongodb che scarica i dati, cosi da evitare tutta la procedura del preprocessing, che, andando a compiere delle misurazioni, è la parte più pesante all'interno dell'intero ciclo di vita del dato.

Si potrebbe aumentare le potenzialità come motore di statistiche, andando a creare nuove metriche e nuovi indici aumentando la varietà di dati da mostrare e la possibilità di rendere la comparazione più completa e realistica. Inoltre un altro modo per rendere i dati sempre più affidabili e completi è allargare il dataset di partenza, aumentando il numero di partite a disposizione e permettendo di avere uno storico più ampio.

Si potrebbe anche andare ad aggiungere il numero di filtri a disposizione rendendo l'applicativo ancora più modulabile, andando a mettere ad esempio dei filtri sulle metriche che compongono gli indici, rendendole selezionabili e de-selezionabili, così da poter visualizzare solo quello che l'utente richiede.

Inoltre si potrebbero inserire degli ulteriori filtri di ricerca basati sui risultati ottenuti, per esempio mostrando solo "i giocatori che hanno una percentuale di Ace maggiore del 20% a partita". Un futuro lavoro che potrebbe essere eseguito è quello di creare un sistema di segnalazioni basato sulla "outlier detection" dei dati.
Partendo dal database creato salvare in maniera asincrona, per ogni giocatore, una tabella con gli indici calcolati raggruppati per diversi attributi, ad esempio calcolando gli indici per ogni superficie, oppure per ogni stagione, creando così un database di riferimento, e salvando questi risultati nel tempo, dare forma ad un modello che li descriva.
Quando viene giocato un match e i punti vengono mandati in real time al database centrale che raccoglie tutti i punti, nel nostro caso la tabella "pointByPoint", gli indici vengono calcolati e vengono restituiti all'applicazione.
L'applicazione poi confronta gli indici appena calcolati con le tabelle "riferimento", eseguendo una "outlier detection".

Outlier nella terminologia statistica viene utilizzato per definire un valore che, in un insieme di misurazioni, è anomalo, ovvero, andando a calcolare le distanze dalle altre osservazioni è distante.
In termini matematici non è esiste una definizione, ma è possibile valutarlo andando a calcolare la distanza dall'intervallo interquartile Q_1,Q_3 di tutte le osservazioni, rapportandola alla misura dell'intervallo stesso.

Si tratta di uno dei rami del Machine Learning e intelligenza artificiale più fertili del momento.
Grazie a questa strategia si riesce quindi a trovare e far risaltare se un particolare dato sta uscendo dalla "norma", ovvero se un giocatore sta compiendo qualcosa di diverso dal solito come le over o under performance.

Questa nuova funzionalità potrebbe rivelarsi uno dei maggiori miglioramenti apportabili all'applicazione poiché si presta a moltissimi utilizzi: ad esempio si potrebbe utilizzare nell'ambito delle piattaforme streaming ed emittenti televisive, dove l'applicativo andrebbe a creare degli avvisi, che collegati ad interfacce grafiche, farebbero comparire dei pop-up visivi a schermo durante una diretta, segnalando la situazione particolare.
Oppure nel caso non si mostrino pop up, l'avviso potrebbe essere comunicato direttamente dal telecronista che avrà l'accesso alle grafiche in tempo reale.
Ed infine ancora nell'ambito delle dirette testuali potrebbe presentarsi un

avviso come "pillola statistica".

In conclusione posso dire che ho creato un prodotto assolutamente funzionante e che con qualche piccolo miglioramento potrebbe essere già messo in commercio, e che, oltretutto, ben si adatta a qualsiasi altro tipo di sport, con i giusti accorgimenti.
Questo progetto mi ha permesso di capire come il mondo dello sport sia permeato di dati e che si possono creare modelli per ogni tipo di evento e sport.

Inoltre mi ha portato a conoscenza della azienda Deltatre, una realtà forse poco conosciuta a livello italiano, ma che è una dei maggiori aziende nell'ambito dell'analisi dati nel mondo sportivo. Ho avuto la possibilità così di conoscere e vedere quali sono gli attori che stanno dietro i calcoli e le statistiche sportive, argomento che da sempre ha destato il mio interesse e la mia curiosità.

Questo progetto quindi mi ha permesso, oltre che approfondire una mia passione, di mettere in campo le mie competenze tecniche maturate durante il mio percorso universitario, mettendomi alla prova su un prodotto vero e proprio che potrebbe realisticamente essere venduto sul mercato; Inoltre permettendomi di affacciarmi al mondo del lavoro misurandomi con richieste, aspettative, scadenze e tempistiche.

Ho avuto l'opportunità sia di affinare il lavoro in gruppo attraverso la collaborazione con i miei colleghi, ma anche di cimentarmi da solo nella risoluzione di situazioni critiche.
Spero un giorno di poter vedere in televisione o in streaming una soluzione del genere poiché conoscendo il meccanismo che sta alla base saprei che il progetto è andato avanti e che si è sviluppato a partire dal lavoro descritto in questa tesi.

Bibliografia

[1] G. Ciaburro, *Le origini del linguaggio pyhton*, 2014, http://www.ciaburro.it/le-origini-del-linguaggio-python/ , 23/12/2021.

[2] Wikipedia contributors, *Python (programming language) — Wikipedia, The Free Encyclopedia*, 2022, https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=1065081237, 23/12/2021.

[3] Streamlit,*Streamlit Documentation*, 2022, https://docs.streamlit.io/, 27/12/2021.

[4] Mastersindatascience,*The role of data science in sports*, 2020, https://www.mastersindatascience.org/resources/big-data-in-sports/, 10/01/2022.

[5] Data-flair,*Data Scientist vs Data Engineer vs Data Analyst – What really differentiates them?*, 2019, https://data-flair.training/blogs/data-scientist-vs-data-engineer-vs-data-analyst/, 11/01/2022.

[6] CONI,*Definizione CONI*, 2020, https://scuoladellosport.coni.it/, 11/01/2022.

[7] Bounous,*Cos'è la performance analysis?*, 2019, https://www.bskilled.it/cose-la-performance-analysis/, 11/01/2021.

[8] Arastey,*What-is-performance-analysis-in-sport*, 2020, https://www.sportperformanceanalysis.com/article/what-is-performance-analysis-in-sport, 11/01/2022.

[9] Hughes, Lipoma, Sibilio,*Performance Analysis*, 2015.

[10] ITF,*ITF Global Tennis Report*, 2019, http://itf.uberflip.com/i/1169625-itf-global-tennis-report-2019-overview/7?, 11/01/2022.

[11] F.Pallone, *Wta-Sap, la tecnologia al servizio del tennis femminile*, Repubblica, 2016.

[12] Infosys, *ATP and Infosys Extend Digital Innovation Partnership*, 2020, https://www.infosys.com/newsroom/press-releases/2020/extend-digital-innovation-partnership.html, 11/01/2022.

[13] F.Femia, *Persone dietro i dati, i maghi dei numeri che animano i match*, La Stampa, 2021.

[14] William Hill News, *Come si gioca a tennis*, 2019, https://williamhillnews.it/tennis/come-si-gioca-a-tennis/, 20/12/2021.

[15] Dotsport, *Fondamentali del tennis*, 2014, https://www.dotsport.it/fondamentali-del-tennis/, 20/12/2021.