Presentazione progetto di Data Management

Costruzione di un dataset per analisi calcistiche

Lorenzo Bruni 886721
Diego Bartoli Geijo 887208

Obiettivo

É possibile individuare una fascia di età di rendimento massimo di un calciatore basandosi sulla nazionalità, le caratteristiche fisiche, il ruolo o il campionato in cui gioca?

Costruiamo un dataset che possa servire come riferimento per rispondere a tali domande.

Data Acquisition

I dati per raggiungere l'obiettivo prefissato sono stati reperiti da tre fonti di dati diverse:

- Understat, dati offensivi dei calciatori (Goal, Assist, Key passes...);
- api-football, dati anagrafici e fisici dei calciatori (Altezza, Peso, Età...);
- Fbref, dati difensivi e di passaggio dei calciatori;

I dati scaricati coprono le stagioni dalla 2017/2018 alla 2020/2021 per i top 5 campionati europei: Premier League (Galles e Inghilterra), Serie A (Italia), La Liga (Spagna), Ligue 1 (Francia), Bundesliga (Germania).

Da Understat e api-football otteniamo un file json per ogni stagione di ogni campionato per un totale di 40 file complessivi. Da Fbref otteniamo due file json per ogni stagione di ogni campionato (uno per i passaggi e uno per le statistiche difensive).

Data Acquisition
Understat

- Tecnica di acquisizione dati API con pacchetto Python **understat**
- Funzione utilizzata **get_league_players:** in input un intero e una stringa, mentre in output una lista di dizionari (dizionario per ogni giocatore)
- Vengono tenuti tutti i dati ad eccezione dell'id Understat e dei giocatori che hanno giocato meno di 15 partite
- In conclusione otteniamo quindi 20 file Json (5 campionati, 4 stagioni)

Data Cleaning
Understat

Sono inserite operazioni di pulizia dati già in fase di acquisizione:

- dati numerici forniti in formato stringa
- giocatori che hanno cambiato squadra a metà stagione e sono rimasti nel solito campionato, presenza doppio valore per campo squadra(percentuale sul totale dei giocatori uguale a 0,03%)
- opzioni per trattare correttamente caratteri speciali
- accorgimento specifico per carattere apostrofo

Data Acquisition
api-football

- Tecnica di acquisizione dati API
- In input 2 stringhe: stagione e id campionato, in più è stata aggiunta una terza stringa indicante la pagina di riferimento
- In output dati per ogni stagione/campionato forniti su più pagine
- Il valore di ritorno è un dizionario di dizionari e liste
- Con una prima richiesta otteniamo il numero di pagine di ogni dizionario, con le altre focus su dizionario players in lista response per scaricare i dati
- In conclusione otteniamo quindi 20 file Json (5 campionati, 4 stagioni)

Data Cleaning

api-football

Sono inserite operazioni di pulizia dati già in fase di acquisizione:

- dati numerici forniti in formato stringa come peso (kg) e altezza (cm)
- campo age indica l'età attuale ma non quella della stagione di riferimento che verrà ricavata da birth date
- eliminazione campi dizionario di non interesse
- opzioni per trattare caratteri speciali

Data Acquisition
Fbref

- Principale vantaggio: disponibilità di statistiche molto moderne e possibilità di ampliare il nostro dataset con statistiche avanzate
- Svantaggi: non disponibilità di un API gratuita, i link non possiedono uno schema fisso adatto per il web scraping, dati disponibili da stagione 2017/2018
- Tecnica utilizzata Web Scraping, viene sfruttata l'url della pagina dove si trova la tabella di interesse
- Dati prelevati: passaggi e azioni difensive dei giocatori, selezionando le metriche che risultano più significative

Data Cleaning

Fbref

Sono inserite operazioni di pulizia dati già in fase di acquisizione:

- dati numerici forniti in formato stringa
- normalizzazione valori colonna Pos e manteniamo solo il primo valore
- I dati sono forniti in formato tabellare ma vengono convertiti in file json per avere una struttura comune, otteniamo quindi 40 file json, 2 per ogni stagione /campionato (file json di passaggio e difensivo)

Data Modelling

80 file, tutti in formato **json**.

Scegliamo il **document based** come modello NoSQL.

Come DBMS scegliamo **MongoDB**.

Per interagire con il database utilizziamo **pymongo**.

Data Modelling

1 collezione per ogni **campionato** e **stagione** dati.

1 documento per ogni **giocatore**.

Struttura:

- name, **age**, nationality, **height**, **weight**, team, **position**
- **general stats:** games, time, red cards, yellow cards
- **offensive stats:** goals, **xG**, **xA**, shots, key passes, npg, npxG, xGChain, xGBuildup
- **defensive stats:** Tkl, **TklW**, Past, Press, **Succ**, **Blocks**, Int
- **passing stats:** Cmp, **Cmp%**, **1 / 3**, PPA, CrsPA, **Prog**

Data integration

Imponiamo in fase di download che **stessi giocatori** abbiano lo **stesso nome** in tutti i dataset.

Il campo che rende possibile l'**aggregazione** è il **nome**.

Non ci sono **conflitti di schema**.

Instance level conflict per il **ruolo** dei giocatori, manteniamo i valori di Fbref.

Carichiamo per primi i file contenenti i giocatori scaricati da api-football, a questi aggreghiamo statistiche offensive (Understat), difensive e di passaggi (Fbref).

Data integration - procedimento

Salviamo i nomi di tutti i **giocatori** scaricati da Understat in una **lista di liste** chiamata **football_players** e le **squadre** in una **lista di liste** chiamata **players_teams**.

Ogni sottolista fa riferimento a una data **stagione** di un certo **campionato**.

Conosciamo ogni sottolista a quale campionato e stagione fa riferimento basandoci sull'**ordine di iterazione**:

- Serie A secondo campionato in ordine di iterazione
- 2019 terza stagione in ordine di iterazione
  ==> Serie A 2019 dodicesima sottolista

La **j-esima squadra** dell'**i-esima sottolista** di **players_teams** è la squadra del **j-esimo giocatore** dell'**i-esima sottolista** di **football_players**.

Data integration - procedimento

Sostituiamo il nome di **ogni giocatore** scaricato da api-football e Fbref con il **nome** dell'opportuna **sottolista** di **football_players** con cui ha **somiglianza massima**. Se non viene trovata **nessuna corrispondenza** il **giocatore** viene **scartato**.

Anche la squadra è utilizzata nel calcolo della somiglianza.

In questo modo **in tutti i dataset** i **nomi** dei giocatori sono quelli di **Understat**.

Tale operazione è implementata con una funzione chiamata **checkname**.

Data integration - checkname

Per ogni giocatore scaricato da **api-football** viene invocata **checkname** che **riceve:**

- la **sottolista** di **players_teams** dell'opportuna stagione e anno
- la **sottolista** di **football_players** dell'opportuna stagione e anno
- il **nome completo** del **giocatore**
- il **nome comune**
- la **squadra**

Considera **ogni stringa** di **football_players**, se la stringa è composta da **due nomi** ne calcola la somiglianza col **nome completo altrimenti** con il **nome comune**.

Per **Fbref** la funzione checkname riceve **solo** il **nome comune**.

Data integration - checkname

La somiglianza viene calcolata con il criterio di **SequenceMatcher**, valore compreso **tra 0 e 1**. Viene **aggiunto 0.2** se la **somiglianza** viene riscontrata **anche** tra i nomi delle **squadre**.

Ritorna la **stringa** di **football_players** con cui si è trovato il **massimo valore di somiglianza**. Se il massimo valore di somiglianza è **minore di 0.8** viene tornata una **stringa vuota**.

Il **valore di ritorno** di checkname viene assegnato al campo **name** del **giocatore**.

Data quality

Accuracy:

- **syntactic** accuracy: nome (Understat), ruolo (Fbref), nazionalità (api-football) **non** presentano **errori sintattici**.

- **semantic** accuracy: **statistiche** di rendimento **moderne** e specifiche per le **tre fasi** di gioco principali.

Data quality

Completeness:

- **0.1%** dei giocatori **non** possiede **passing stats** e **defensive stats**, sono i giocatori presenti nei dati scaricati da api-football ma non in quelli scaricati da Fbref.

- ipotesi di **mondo aperto**, assumendo Understat come rappresentazione completa del mondo reale otteniamo una **object completeness media** tra le collezioni del **92%**.

Data quality

Consistency:

- anche a **costo** di **perdere** qualche **giocatore** abbiamo imposto che i **nomi** dei **giocatori** fossero quelli di **Understat** in **tutti i file** scaricati.

- **perdiamo** in **completeness** per **massimizzare** la **consistency**.

Grazie per l’attenzione.

Lorenzo Bruni
Diego Bartoli Geijo