Report for Python

Football Statistics Scraper

Overview

The Python program is designed to scrape statistical data for football players from the 2024-2025 English Premier League season on fbref.com. It uses Selenium and BeautifulSoup for web scraping, processes the data, and saves the results to a file named results.csv.

Key Features

Data Collection

1. Accesses various data tables from fbref.com (general stats, goalkeeping, shooting, passing, goal creation, defensive actions, possession, and miscellaneous stats).
2. Filters players with more than 90 minutes of playtime.
3. Extracts metrics such as nationality, team, position, age, matches played, goals, assists, cards, and advanced stats (xG, xAG, PrgC, SCA, etc.).
2.

Data Processing:

1. Standardizes column names, handles duplicates, and maps them to required fields.
2. Converts numeric data, fills missing values with "N/a".
3. Merges data from multiple tables using Player_ID.
4. Sorts players alphabetically by first name and cleans data (e.g., Nation column).

Output:

1. Saves data to results.csv with a predefined column structure.
2. Uses UTF-8 encoding to support special characters.

Strengths

- **Automation**: Fully automates data scraping and processing from multiple tables.
- **Error Handling**: Robust checks for page access, table parsing, and missing values.
- **Clear Structure**: Uses a data_tables dictionary for table configurations, making it easy to extend.

Limitations

- **Web Structure Dependency**: Changes to fbref.com's layout or table IDs may break the script.
- **Performance**: Selenium in headless mode can be slow for large datasets.
- **No Parallelization**: Lacks concurrent processing to speed up data fetching.

Code Explanation

Main Functions and Components

Configuration (data_tables**)**:

- A dictionary defining the endpoints, fields, and field mappings for each table (e.g., general, goalkeeping, shooting).
- **Purpose**: Organizes the scraping process by mapping fbref.com table endpoints to desired statistics and their output names.
- **Example**: For the "general" table, it maps fields like Gls to Goals and xG.1 to xG/90.

fetch_table_data(table_config):

- **Purpose**: Fetches a specific table from fbref.com using Selenium and processes it into a pandas DataFrame.
- **Key Steps**:

        - Configures a headless Chrome browser with user-agent headers to mimic a real user.
        - Navigates to the table's URL (e.g., https://fbref.com/en/comps/9/stats/Premier-League-Stats).
        - Waits for the table to load using WebDriverWait and locates it by ID or class.
        - Uses BeautifulSoup to parse the table's HTML and extract player IDs from links.
        - Converts the table to a DataFrame using pd.read_html.
        - Handles column renaming, duplicate columns, and missing data by mapping columns to expected fields and filling gaps with "N/a".
        - Converts numeric columns and cleans specific fields like Age.
    - **Output**: A DataFrame with cleaned data for the specified table, including Player_ID for merging.

Data Merging and Aggregation:

- **Purpose**: Combines data from all tables and aggregates statistics for each player.
- **Key Steps**:
    - Starts with the general table (if available) as the base DataFrame, filtering for players with >90 minutes.
    - Merges other tables (e.g., goalkeeping, passing) on Player_ID using pd.merge with a left join.
    - Groups data by Player_ID and applies aggregation rules (e.g., sum for Goals, mean for Gls/90) defined in aggregation_rules.
    - Sorts the final DataFrame by the player's first name.
- **Output**: A unified DataFrame with all required statistics.

Data Cleaning and Output:

- **Purpose**: Finalizes the DataFrame and saves it to results.csv.
- **Key Steps**:

        - Fills missing values with "N/a".
        - Ensures all required columns are present, adding "N/a" for missing ones.
        - Cleans the Nation column to extract uppercase country codes (e.g., "ENG" from "eng ENG").
        - Checks data quality by flagging columns with excessive "N/a" values (>50% of rows).
        - Saves the DataFrame to results.csv with UTF-8 encoding.
    - **Output**: A CSV file with the structured, cleaned data.

Conclusion

The program effectively meets the requirements for scraping and processing football player statistics, producing a well-structured CSV output. Improvements could include parallelizing data fetching or integrating an API (if available) for better performance. Regular maintenance is needed to ensure compatibility with fbref.com’s structure.