Top 5 Leagues goals by nationality

October 15, 2024

0.1 Data Parsing: Top 5 Football Leagues Historical Goalscorers by Nation

Ever wondered the amount of goals that players from your country have scored in Europe Top 5 Football Leagues? Well, thanks to powerful libraries such as BeautifulSoup and Pandas, this is possible.

All the data is collected from https://www.worldfootball.net/goalgetter/.

*The data begins in the season of 1963-1964 because this was the year where Bundesliga was founded. Therefore, it would be unfair to consider previous years.

Import the necessary libraries

[1]:
```
# import libraries for data manipulation
import numpy as np
import pandas as pd

# import libraries for parsing
import requests
from bs4 import BeautifulSoup

# to suppress warnings
import warnings
warnings.filterwarnings('ignore')
```

Helper function to generate text for the columns having two dates as parameters.

[2]:
```
def generate_seasons_years(from_date, to_date):
    seasons_text = []
    for year in range(from_date, to_date):
        seasons_text.append(str(year) + "-" + str(year + 1))
    return seasons_text
```

Recursive function that fills the goals into the dictionary per season.

[3]:
```
def fill_data(global_dict, country, season, goals,␣
 ↪empty_seasons_dictionary=None):
    if (country in global_dict.keys()):
        global_dict[country][season] += goals
        return
    else: # In case this is the first time that a country appears, firs it is␣
↪initialized and then filled through recursion.
        global_dict[country] = empty_seasons_dictionary.copy()
        fill_data(global_dict, country, season, goals)
```

Function that parses the webpages. Then it extracts the relevant keywords to populate the dictioanry

[4]:
```
def parse_and_fill(global_dict, url, season, empty_seasons_dictionary):
    # Fetch the webpage content
    response = requests.get(url)

    # Parse the HTML using BeautifulSoup
    soup = BeautifulSoup(response.content, 'html.parser')

    # Locate the table containing the data (goal scorers, etc.)
    table = soup.find('table', class_='standard_tabelle')  # Look for the␣
↪specific class used in the table (in case there's multiple)

    # Extract the data
    rows = table.find_all('tr')
    for row in rows[1:]:  # Skip the header row
        cols = row.find_all('td')
        cols = [col.text.strip() for col in cols]  # Clean the text
        # Save important data
        country = cols[3]
        goals = int(cols[5].rsplit(" ")[0])
        # Use the data to populate the dictionary
        fill_data(global_dict, country, season, goals, empty_seasons_dictionary)
```

Function that produces the right url. Some webpages have a non-intuitive webpage, therefore some if statements are introduced

[5]:
```
def get_urls(league, season):
    # Base website url
    base_url = 'https://www.worldfootball.net/goalgetter/'
    urls = []

    # Checks for specific cases
    if (league == "esp-primera-division" and season == "2016-2017"):
        urls.append(base_url + league + "-" + season + "_2/")
    elif (league == "esp-primera-division" and season == "1986-1987"):
        spain_leagues = ["esp-primera-division-1986-1987-playoff-1-6",␣
↪"esp-primera-division-1986-1987-playoff-13-18",␣
↪"esp-primera-division-1986-1987-playoff-7-12",␣
↪"esp-primera-division-1986-1987-vorrunde"]
```

```
        # In this season, the league was divided in sub-leagues. This for loop␣
↪makes sure that all of them are included.
        for spain_league in spain_leagues:
            urls.append(base_url + spain_league + "/")
    else:
        # The most common case.
        urls.append(base_url + league + "-" + season + "/")
    return urls
```

Function that intializes a dictionary row with 0s as values.

[6]:
```
def create_empty_seasons_dictionary(seasons):
    # Empty dictionary is defined
    seasons_dictionary = {}
    # Populate the dictionary
    for season in seasons:
        seasons_dictionary[season] = 0
    return seasons_dictionary
```

Main Function that iterates over each season and each league and populates the dictionary using helper functions.

[7]:
```
def extract_values_top_5_leagues(from_date, to_date):
    # Sets the relevant parameters for the iterations
    goals_per_nation_and_year = {}
    seasons = generate_seasons_years(int(from_date), int(to_date))
    empty_seasons_dictionary = create_empty_seasons_dictionary(seasons)
    leagues = ["eng-premier-league", "fra-ligue-1", "bundesliga",␣
↪"ita-serie-a", "esp-primera-division"]

    # Main loop that iterates through every leaguer per each season.
    for season in seasons:
        for league in leagues:
            urls = get_urls(league, season)
            for url in urls: # for the case where there are multiple urls in␣
↪one league in a single season
                parse_and_fill(goals_per_nation_and_year, url, season,␣
↪empty_seasons_dictionary)

    #Returns a sorted dictionary based on the name of the keys.
    return dict(sorted(goals_per_nation_and_year.items()))
```

Calls the main function

[8]:
```
# Main dictionary produced by the program stored in a variable
final_dictionary = extract_values_top_5_leagues(1963,2024)
```

[9]:
```python
# Check if an empty key exists and delete it if so
if "" in final_dictionary:
    del final_dictionary[""]

### final_dictionary
```

0.1.1 Creation of CSV/Excel file.

Initialize a list where the dictionary will be transformed.

[10]:
```python
list_for_csv = []
# Name for the outer keys stored in the header
headers = ["Countries"]
for country, inner_dict in final_dictionary.items():
    for key in inner_dict.keys():
        # NAmes of the inner keys (seasons) stored in the header
        headers.append(key)

    # Just add it once
    break
```

Now populate the list with the correct format

[11]:
```python
# Loop that iterates over the inner dictionary items
for country, inner_dict in final_dictionary.items():
    # Creates a row with the country as its first value
    country_goals = [country]
    for value in inner_dict.values():
        # Appends the goals per season in the right order
        country_goals.append(int(value))
    list_for_csv.append(country_goals)
```

Makes the necessary arrangements to convert it into a dataframe

[12]:
```python
# Saves the python list as a numpy array
list_as_numpy_array = np.array(list_for_csv)
# Creates the dataframe
df = pd.DataFrame(list_as_numpy_array, columns=headers)
# Forces numerical value
df.iloc[:, 1:] = df.iloc[:, 1:].apply(pd.to_numeric)
# Creates a column that accumulates all the goals per country
df['sum'] = df.iloc[:, 1:].sum(axis=1)
# Sorts the dataframe by cumulative total sum.
df = df.sort_values(by="sum", ascending = False)
# Resets index to assure proper display
df.reset_index(drop = True, inplace=True)

# Saves the file as CSV or Excel
```

```

### df.to_csv('top_5_leagues_countries_cumulative.csv', index=False)
### df.to_excel('top_5_leagues_countries_cumulative.xlsx', index=False)
```