

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CNTT**

----------

**BÁO CÁO BÀI**

**TẬP LỚN PYTHON**

*Đơn vị:*

***Giảng viên hướng dẫn:*** ***Kim Ngọc Bách***

***Sinh viên thực hiện: Hoàng Đình Nhật Văn***

***Mã sinh viên: B22DCCN888***

***Nhóm: 11***

**Hà Nội, 2024**

# I.(4đ)

Viết chương trình Python thu thập dữ liệu phân tích cầu thủ với yêu cầu như sau:

* + Thu thập dữ liệu thống kê [\*] của tất cả các cầu thủ có số phút thi đấu nhiều hơn 90 phút tại giải bóng đá ngoại hạng Anh mùa 2023-2024.
  + Nguồn dữ liệu: https://fbref.com/en/
  + Ghi kết quả ra file 'results.csv', bảng kết quả có cấu trúc như sau:

+ Mỗi cột tương ứng với một chỉ số.

+ Thứ tự các cầu thủ sắp xếp theo thứ tự tên (First Name), nếu trùng tên thì xếp theo độ tuổi từ lớn đến nhỏ.

+ Chỉ số nào không có hoặc không áp dụng thì để là N/a

**Bài làm:**

#### 1. Mục tiêu của bài tập

Mục tiêu của bài tập này là thu thập và xử lý dữ liệu thống kê của các cầu thủ tham gia mùa giải Premier League 2023-2024 từ trang fbref.com. Từ đó, chỉ lấy các cầu thủ có thời gian thi đấu lớn hơn 90 phút, lưu dữ liệu dưới dạng file CSV và làm cơ sở cho các phân tích chuyên sâu về đội bóng và cầu thủ sau này.

#### 2. Công cụ và thư viện sử dụng

Sử dụng Python làm ngôn ngữ lập trình chính cho bài tập này, kết hợp với các thư viện sau:

* **Pandas**: Thư viện này giúp xử lý và tổ chức dữ liệu dưới dạng bảng, rất tiện lợi cho các thao tác làm sạch và phân tích dữ liệu.
* **Selenium**: Được sử dụng để thu thập dữ liệu từ trang web fbref.com. Requests phù hợp cho việc lấy dữ liệu tĩnh, trong khi Selenium hỗ trợ các trang có nội dung động.
* **CSV**: File kết quả được lưu dưới định dạng CSV giúp dễ dàng truy xuất và sử dụng trong các ứng dụng phân tích khác.
* **Beautiful Soup**: một thư viện Python được dùng để phân tích cú pháp HTML và XML, giúp dễ dàng trích xuất dữ liệu từ các trang web. Nó cho phép bạn điều hướng, tìm kiếm và sửa đổi nội dung HTML một cách hiệu quả.

#### 3. Quy trình thực hiện và giải thích chi tiết từng đoạn mã

##### *Bước 1: Thu thập dữ liệu từ https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats*

**Mã Python:**

from selenium import webdriver

from bs4 import BeautifulSoup

import pandas as pd

driver = webdriver.Chrome()

driver.get('https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats')

driver.implicitly\_wait(10)

html = driver.page\_source

soup = BeautifulSoup(html, 'html.parser')

**Giải thích**:

* **driver = webdriver.Chrome**(): Khởi tạo một phiên bản trình duyệt Chrome.

##### driver.get(...): Dùng để mở trang fbref chứa các số liệu thống kê Premier League mùa 2023-2024.

* **driver.implicitly\_wait(10):** Chỉ định thời gian chờ đợi tối đa là 10 giây để trang tải và các phần tử trên trang sẵn sàng được tương tác. Điều này giúp đảm bảo rằng các thành phần HTML cần thiết được tải đầy đủ trước khi tiếp tục xử lý.
* **html = driver.page\_source:** Lấy mã HTML của trang hiện tại mà trình duyệt Chrome đã tải.
* **soup = BeautifulSoup(html, 'html.parser'):** Tạo một đối tượng Beautiful Soup, giúp phân tích cú pháp HTML để dễ dàng trích xuất các phần tử HTML cần thiết trong các bước tiếp theo.

*Bước 2: Thực hiện việc lấy các tiêu đề (headers) của bảng dữ liệu "stats\_standard" từ trang fbref.*

**Các bước thực hiện:**

**-Tìm bảng có ID “stats\_standard”:**

table = soup.find("table", {'id':'stats\_standard'})

* **soup.find("table", {'id':'stats\_standard'**}): Tìm phần tử <table> có thuộc tính id là "stats\_standard", tức là bảng dữ liệu chính chứa các số liệu thống kê của cầu thủ.

**-**Lấy và kết hợp các tiêu đề từ hai hàng trong phần đầu của bảng (header) để tạo ra một danh sách tiêu đề đầy đủ cho bảng dữ liệu:

headers\_1 = [th.get\_text() for th in table.find('thead').find\_all('tr')[0].find\_all('th')]

headers\_2 = [th for th in table.find('thead').find\_all('tr')[1].find\_all('th')]

headers = []

for h2 in headers\_2:

    check = 0

    for h1 in headers\_1:

        if h1 == h2.get('data-over-header') :

            headers.append(f'{h1} {h2.get\_text()}')

            check = 1

    if check == 0:

        headers.append(h2.get\_text())

* **headers\_1**: Lấy nội dung văn bản (get\_text()) của tất cả các phần tử <th> trong hàng đầu tiên ([0]) của phần thead (đầu bảng). Các giá trị này đại diện cho nhóm tiêu đề chung, như “Player”, “Nation”, “Squad”, ”Playing Time”.
* **headers\_2**: Lấy tất cả phần tử <th> trong hàng thứ hai ([1]) của thead. Mỗi phần tử này là một tiêu đề chi tiết, thường nằm dưới các tiêu đề nhóm.
* Vòng lặp “for h1 in headers\_1” kiểm tra xem giá trị “data-over-header” của h2 có khớp với tiêu đề nhóm nào h1.
* Nếu khớp, tiêu đề nhóm và tiêu đề chi tiết sẽ được kết hợp thành dạng "{h1} {h2.get\_text()}" rồi thêm vào headers.
* check = 1 đánh dấu rằng tiêu đề đã được ghép.
* Nếu không tìm thấy nhóm tương ứng (check == 0), chỉ lấy tiêu đề chi tiết của h2 và thêm vào headers.

##### *Bước 3: Lấy dữ liệu từ bảng và lọc các cầu thủ có thời gian thi đấu > 90 phút*

**Mã Python**:

player\_data = []

for row in table.find('tbody').find\_all('tr'):

    if row.find('td', {'data-stat': 'player'}):

        player = [td.text if td.text.strip() else "0" for td in row.find\_all('td')]

        minutes\_played\_text = player[8]

        minutes\_played = int(minutes\_played\_text.replace(',', '')) if minutes\_played\_text else 0

        player[8] = minutes\_played

        if minutes\_played > 90:

            player\_data.append(player)

**Giải thích**:

##### **for row in table.find('tbody').find\_all('tr')** với điều kiện **row.find('td', {'data-stat': 'player'}):** Tìm tất cả các hàng có thẻ <tr> sao cho tồn tại “data-stat” = “player” (để loại trừ các hàng không phải là cầu thủ).

* **player = [td.text if td.text.strip() else "0" for td in row.find\_all('td')]:** Lấy dữ liệu cầu thủ từ các ô trong văn bản dưới dạng văn bản.

##### **minutes\_played\_text = player[8]; minutes\_played = int(minutes\_played\_text.replace(‘,’, ‘’)) if minutes\_played\_text else 0; player[8] = minutes\_played:** lấy ô “Playing Time Min” của player dưới dạng text rồi định dạng nó về dạng số nguyên và gán lại cho player[8].

##### **if minutes\_played > 90: player\_data.append(player):** Lọc cầu thủ có số phút thi đấu > 90 và thêm vào “player\_data”.

##### *Bước 4: Chuyển dữ liệu cầu thủ thành DataFrame*

**Mã Python:**

dfstats = pd.DataFrame(player\_data, columns=headers[1:])

dfstats['Age'] = pd.to\_numeric(dfstats['Age'], errors='coerce')

dfstats = dfstats.sort\_values(by=['Player', 'Age'], ascending=[True, False])

dfstats = dfstats.drop(columns=['Matches'])

**Giải thích**:

* **dfstats = pd.DataFrame(player\_data, columns=headers[1:]):** Tạo một DataFrame dfstats từ danh sách player\_data, với các cột lấy từ headers (bắt đầu từ vị trí thứ 2, bỏ qua cột “Rk”).
* **dfstats['Age'] = pd.to\_numeric(dfstats['Age'], errors='coerce'):** Chuyển giá trị trong cột "Age" thành kiểu số. Nếu có giá trị không thể chuyển đổi (vd. trống hoặc không phải số), hàm sẽ gán giá trị NaN vào.
* **dfstats = dfstats.sort\_values(by=['Player', 'Age'], ascending=[True, False]):** Sắp xếp dfstats theo cột "Player" (tăng dần) và "Age" (giảm dần).
* **dfstats = dfstats.drop(columns=['Matches']):** Xóa cột "Matches" vì nó không còn cần thiết cho phân tích.

##### *Bước 5: Nối bảng dfstats với bảng Keeper*

**Mã Python:**

driver.get('https://fbref.com/en/comps/9/2023-2024/keepers/2023-2024-Premier-League-Stats')

driver.implicitly\_wait(10)

html = driver.page\_source

soup = BeautifulSoup(html, 'html.parser')

table\_keeper = soup.find("table", {'id':'stats\_keeper'})

headers\_1 = [th.get\_text() for th in table\_keeper.find('thead').find\_all('tr')[0].find\_all('th')]

headers\_2 = [th for th in table\_keeper.find('thead').find\_all('tr')[1].find\_all('th')]

headers\_keeper = []

for h2 in headers\_2:

    check = 0

    for h1 in headers\_1:

        if h1 == h2.get('data-over-header') :

            headers\_keeper.append(f'Keeper {h1} {h2.get\_text()}')

            check = 1

    if check == 0:

        headers\_keeper.append(h2.get\_text())

player\_data = []

for row in table\_keeper.find('tbody').find\_all('tr'):

    if row.find('td', {'data-stat': 'player'}):

        player = [td.text if td.text.strip() else "0" for td in row.find\_all('td')]

        player\_data.append(player)

df\_keeper = pd.DataFrame(player\_data, columns=headers\_keeper[1:])

df\_keeper = df\_keeper.drop(columns=['Age', 'Keeper Playing Time MP', 'Keeper Playing Time Starts', 'Keeper Playing Time Min', 'Keeper Playing Time 90s', 'Matches'])

df\_merged\_stats = pd.merge(dfstats, df\_keeper, on=['Player','Nation', 'Pos', 'Squad', 'Born'], how='left')

**Giải thích**:

* Thực hiện các bước như 1,2,3,4 để lấy được DataFrame bảng Keeper.
* Loại bỏ các cột 'Age', 'Keeper Playing Time MP', 'Keeper Playing Time Starts', 'Keeper Playing Time Min', 'Keeper Playing Time 90s', và 'Matches' từ DataFrame df\_keeper. Những cột này có thể không còn cần thiết cho phân tích vì đã được xử lý ở dfstats.
* Kết hợp(merge) hai DataFrames dfstats và df\_keeper dựa trên các cột chung 'Player', 'Nation', 'Pos', 'Squad', và 'Born', how='left' chỉ định kiểu kết hợp là "left join", nghĩa là tất cả các dòng từ dfstats sẽ được giữ nguyên, và dữ liệu từ df\_keeper sẽ được thêm vào các dòng có cùng giá trị ở các cột kết hợp. Những cầu thủ không có dữ liệu trong df\_keeper sẽ có giá trị NaN trong các cột thêm vào từ df\_keeper.

##### *Bước 6: Làm các bước tương tự bước 5 thu thập dữ liệu từ các bảng “Shooting”, “Passing”, “Passing Types”, “Goad and Shot Creation”, “Defensive Actions”, “Possesstion”, “Playing Time”, “Miscellaneous Stats”*

##### *Bước 7: Thêm “N/a” vào các ô không có giá trị và lưu bảng vào “result.csv”*

**Mã Python:**

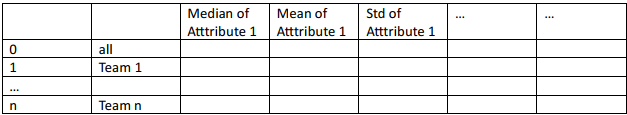
df\_merged\_stats = df\_merged\_stats.fillna('N/a')

df\_merged\_stats.to\_csv('result.csv')

**II. (2đ)**

- Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số.

- Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ trong toàn giải và của mỗi đội. Ghi kết quả ra file results2.csv, format như sau:



- Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội.

- Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024

- Histogram Plot: <https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html>

**Bài làm:**

**1. Mục tiêu**

Mục tiêu của đoạn mã này là phân tích dữ liệu thống kê cầu thủ và đội bóng từ mùa giải 2023-2024 với các nhiệm vụ sau:

1. Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số.
2. Tính trung vị, trung bình, và độ lệch chuẩn của mỗi chỉ số cho toàn giải và từng đội, sau đó lưu kết quả vào file results2.csv.
3. Vẽ biểu đồ histogram để trực quan hóa sự phân bố của mỗi chỉ số cho từng đội và toàn giải.
4. Xác định đội bóng có điểm số trung bình cao nhất ở mỗi chỉ số và tìm đội có phong độ tốt nhất giải đấu dựa trên các chỉ số đó.

### 2. Các Thư viện Sử Dụng

* **Pandas**: Thư viện hỗ trợ thao tác và phân tích dữ liệu.
* **Matplotlib**: Thư viện hỗ trợ vẽ biểu đồ và trực quan hóa dữ liệu.

### 3. Tóm tắt Dữ liệu

Dữ liệu được lưu trữ trong file result.csv chứa các thông tin về cầu thủ, bao gồm tên, tuổi, đội bóng, quốc gia, vị trí chơi, cùng các chỉ số thống kê khác. Dữ liệu này sẽ được đọc và xử lý thông qua Pandas.

### 4. Phân tích Chi Tiết Từng Phần Mã

#### *a. Lấy dữ liệu từ bảng result.csv.*

**Mã Python:**

df = pd.read\_csv('result.csv')

attributes = df.columns[7:]

**Giải thích:**

* Sử dụng Pandas để đọc file result.csv và lưu dữ liệu vào DataFrame df
* Dùng biến “attributes” để lưu danh sách các cột chỉ số từ thứ 8 trở đi

#### *b. Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số.*

**Bước 1: Chuẩn bị dữ liệu cho các cột**

**Mã Python:**

top3\_results = []

for col in df.columns[2:]:

    df[col] = pd.to\_numeric(df[col], errors='coerce').fillna(0)

**Giải thích:**

* Chuyển đổi các cột từ cột thứ ba trở đi(bỏ qua cột tên và cột thứ tự) sang kiểu số.
* **pd.to\_numeric(df[col], errors='coerce'):** Chuyển đổi giá trị trong mỗi cột sang kiểu số. Nếu giá trị nào không chuyển đổi được (ví dụ, chuỗi ký tự), nó sẽ được chuyển thành NaN.
* **.fillna(0):** Thay thế các giá trị NaN bằng 0.

**Bước 2: Lặp qua các chỉ số và tìm top 3 giá trị cao nhất và thấp nhất**

**Mã Python:**

for attr in attributes:

    top3\_highest = df.nlargest(3, attr)[['Player', attr]]

**Giải thích:**

* **for attr in attributes**: Duyệt qua từng chỉ số trong danh sách attributes.
* **top3\_highest = df.nlargest(3, attr)[['Player', attr]]**: Tìm top 3 cầu thủ có giá trị cao nhất cho chỉ số hiện tại (attr) và lưu lại tên cầu thủ cùng với giá trị của chỉ số đó.

**Bước 3: Lưu top 3 cầu thủ có giá trị cao nhất**

**Mã Python:**

    for \_, row in top3\_highest.iterrows():

        top3\_results.append({

            'Attribute': attr,

            'Type': 'Highest',

            'Player': row['Player'],

            'Value': row[attr]

        })

**Giải thích: Duyệt qua từng hàng trong top3\_highest**: Với mỗi cầu thủ trong top 3 cao nhất, thêm một từ điển chứa thông tin vào danh sách top3\_results:

* **'Attribute':** Chỉ số hiện tại (attr)
* **'Type':** Ghi chú đây là Highest (cao nhất).
* **'Player':** Tên cầu thủ.
* **'Value':** Giá trị của chỉ số cho cầu thủ đó.

**Bước 4: Tương tự top 3 cầu thủ có giá trị thấp nhất**

**Mã Python:**

    top3\_lowest = df.nsmallest(3, attr)[['Player', attr]]

    for \_, row in top3\_lowest.iterrows():

        top3\_results.append({

            'Attribute': attr,

            'Type': 'Lowest',

            'Player': row['Player'],

            'Value': row[attr]

        })

**Giải thích:**

* **top3\_lowest = df.nsmallest(3, attr)[['Player', attr]]**: Tìm top 3 cầu thủ có giá trị thấp nhất cho chỉ số hiện tại.

**Bước 5: Lưu kết quả vào file CSV**

**Mã Python:**

top3\_df = pd.DataFrame(top3\_results)

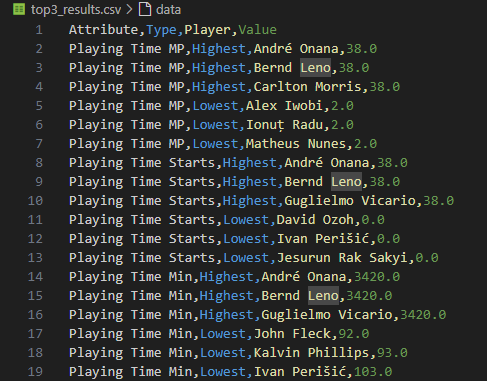
top3\_df.to\_csv('top3\_results.csv', index=False)

**Giải thích:**

* **top3\_df = pd.DataFrame(top3\_results)**: Tạo một DataFrame top3\_df từ danh sách top3\_results.

### top3\_df.to\_csv('top3\_results.csv', index=False): Ghi DataFrame vào file top3\_results.csv mà không lưu chỉ số hàng (index=False).

**Kết quả mẫu (chi tiết các kết quả ở file “top3\_result.csv”):**

****

#### *c. Tính trung vị, trung bình, và độ lệch chuẩn của mỗi chỉ số cho từng đội và toàn giải.*

**Bước 1: Khởi tạo đội bóng và thống kê**

**Mã Python:**

teams = df['Squad'].unique()

stats = ['Median', 'Mean', 'Std']

columns = ['Team']

**Giải thích:**

* **teams = df['Squad'].unique():** Lấy danh sách các đội bóng duy nhất từ cột 'Squad'.
* **stats = ['Median', 'Mean', 'Std']:** Các thống kê cần tính toán: trung vị, trung bình, độ lệch chuẩn.
* **columns = ['Team']:** Khởi tạo danh sách cột với tên cột đầu tiên là ‘Team’.

**Bước 2: Tạo cột DataFrame cho kết quả**

**Mã Python:**

for col in attributes:

    for stat in stats:

        columns.append(f'{stat} of {col}')

result\_df = pd.DataFrame(columns=columns)

**Giải thích:** Với mỗi chỉ số (trong attributes), tạo ra các cột mới theo tên các thống kê vào danh sách cột “columns”, ví dụ: “Median of Playing Time MP”, “Mean of Playing Time MP”, “Std of Playing Time MP”. Sau đó tạo DataFrame “result\_df” với danh sách các cột bằng “columns”

**Bước 3:** **Tính thống kê cho toàn bộ giải đấu**

**Mã Python:**

team\_data = ['all']

for col in attributes:

    numeric\_col = pd.to\_numeric(df[col], errors='coerce')

    team\_data.append(numeric\_col.median())

    team\_data.append(numeric\_col.mean())

    team\_data.append(numeric\_col.std())

result\_df.loc[0] = team\_data

**Giải thích:**

* **team\_data = ['all']:** Tạo 1 danh sách với cột đầu tiên tên đội là “all” cho toàn giải đấu.
* **for col in attributes:** Lặp qua tất cả các chỉ số đã lấy
* **numeric\_col = pd.to\_numeric(df[col], errors='coerce'):** Chuyển đổi các giá trị của các ô thành dạng số nếu không phải là số thì chuyển về dạng NaN.
* **team\_data.append(numeric\_col.median()):** Trung vị của cột.
* **team\_data.append(numeric\_col.mean()):** Trung bình của cột.
* **team\_data.append(numeric\_col.std()):** Độ lệch chuẩn của cột.
* **result\_df.loc[0] = team\_data:** team\_data là danh sách chứa các kết quả cho toàn bộ giải đấu (all), lưu kết quả vào hàng đầu tiên (loc[0]) của result\_df.

**Bước 4:** **Tính thống kê cho từng đội bóng**

**Mã Python:**

for i, team in enumerate(teams):

    team\_data = [team]

    team\_df = df[df['Squad'] == team]

    for col in attributes:

        numeric\_col = pd.to\_numeric(team\_df[col], errors='coerce')

        team\_data.append(numeric\_col.median())

        team\_data.append(numeric\_col.mean())

        team\_data.append(numeric\_col.std())

    result\_df.loc[i+1] = team\_data

**Giải thích:**

* **for i, team in enumerate(teams):** Lọc qua tất cả các đội với biến i tăng dần từ 0.
* **team\_data = [team]:** tạo 1 danh sách với tên cột đầu tiên là tên đội bóng đang xét.
* **team\_df = df[df['Squad'] == team]:** tạo 1 bảng mới lọc với điều kiện “Squad” là team đang xét.
* Thực hiện như bước 2 để lấy các giá trị trung vị, trung bình và độ lệnh chuẩn của bảng “team\_df”.
* **result\_df.loc[i+1] = team\_data:** team\_data là danh sách chứa các kết quả cho từng đội, lưu kết quả vào hàng thứ i+1 của result\_df.

**Bước 5:** **Lưu kết quả vào file result2.csv**

**Mã Python:**

result\_df.to\_csv('results2.csv')

*d.Vẽ biểu đồ Histogram cho mỗi chỉ số cho các cầu thủ toàn giải và mỗi đội.*

**Bước 1: Vẽ biểu đồ histogram cho tất cả cầu thủ (theo từng thuộc tính)**

**Mã Python:**

save\_path = f"D:/CodePython/Histograms"

for attr in attributes:

    plt.figure(figsize=(10, 6))

    df[attr] = pd.to\_numeric(df[attr], errors='coerce')

    df[attr].dropna().hist(bins=30, color='blue', alpha=0.7)

    plt.title(f'Histogram of {attr} for all players')

    plt.xlabel(attr)

    plt.ylabel('Frequency')

    plt.grid(False)

    safe\_attr = attr.replace("/", "\_")

    plt.savefig(f'{save\_path}/Allplayers {safe\_attr} Histogram.png')

    plt.show()

**Giải thích:**

* **save\_path = f"D:/CodePython/Histograms"**: Đường dẫn lưu trữ các tệp hình ảnh histogram là thư mục Histograms.
* **for attr in attributes:**: Vòng lặp để duyệt qua tất cả các thuộc tính (cột) trong danh sách attributes. Mỗi lần lặp sẽ vẽ một histogram cho thuộc tính đó.
* **plt.figure(figsize=(10, 6))**: Tạo một hình mới với kích thước 10x6 inch.
* **df[attr] = pd.to\_numeric(df[attr], errors='coerce')**: Chuyển cột “attr” sang dạng số (numeric). Nếu có giá trị không phải số, nó sẽ chuyển thành NaN.
* **df[attr].dropna().hist(bins=30, color='blue', alpha=0.7)**: Vẽ biểu đồ histogram cho cột attr, bỏ qua các giá trị NaN. Biểu đồ có 30 cột, màu xanh dương, và độ trong suốt là 0.7.
* **plt.title(f'Histogram of {attr} for all players')**: Thiết lập tiêu đề cho biểu đồ, hiển thị tên của thuộc tính hiện tại cho tất cả cầu thủ.
* **plt.xlabel(attr) và plt.ylabel('Frequency')**: Gắn nhãn cho trục X với tên thuộc tính và trục Y với Frequency.
* **plt.grid(False)**: Tắt lưới trên biểu đồ.
* **safe\_attr = attr.replace("/", "\_")**: Thay thế ký tự / trong tên thuộc tính bằng \_ để đảm bảo tên tệp không bị lỗi khi lưu.
* **plt.savefig(f'{save\_path}/Allplayers {safe\_attr} Histogram.png')**: Lưu biểu đồ dưới dạng tệp ảnh PNG vào save\_path. Tên tệp bao gồm “Allplayers”, tên thuộc tính (safe\_attr) và “Histogram”.
* **plt.show()**: Hiển thị biểu đồ.

**Bước 2: Vẽ biểu đồ histogram cho từng đội bóng (theo từng thuộc tính)**

**Mã Python:**

for team in teams:

    team\_df = df[df['Squad'] == team]

    for attr in attributes:

        plt.figure(figsize=(10, 6))

        team\_df[attr] = pd.to\_numeric(team\_df[attr], errors='coerce')

        team\_df[attr].dropna().hist(bins=20, color='green', alpha=0.7)

        plt.title(f'Histogram of {attr} for {team}')

        plt.xlabel(attr)

        plt.ylabel('Frequency')

        plt.grid(False)

        safe\_attr = attr.replace("/", "\_")

        plt.savefig(f'{save\_path}/{team} {safe\_attr} Histogram.png')

        plt.show()

**Giải thích:**

* **for team in teams:** Vòng lặp để duyệt qua từng đội bóng trong danh sách teams.
* **team\_df = df[df['Squad'] == team]**: Lọc dữ liệu của DataFrame df chỉ lấy các cầu thủ thuộc đội bóng hiện tại (team) và lưu vào team\_df.
* **for attr in attributes:**: Vòng lặp để duyệt qua từng thuộc tính (cột) trong danh sách attributes. Mỗi lần lặp sẽ vẽ một histogram cho thuộc tính đó của đội bóng hiện tại.
* **plt.figure(figsize=(10, 6))**: Tạo một hình mới với kích thước 10x6 inch.
* **team\_df[attr] = pd.to\_numeric(team\_df[attr], errors='coerce')**: Chuyển cột attr trong team\_df sang dạng số, chuyển các giá trị không phải số thành NaN.
* **team\_df[attr].dropna().hist(bins=20, color='green', alpha=0.7)**: Vẽ histogram cho cột attr trong team\_df, bỏ qua các giá trị NaN. Biểu đồ có 20 cột, màu xanh lá cây, và độ trong suốt là 0.7.
* **plt.title(f'Histogram of {attr} for {team}')**: Thiết lập tiêu đề cho biểu đồ, hiển thị tên thuộc tính và đội bóng hiện tại.
* **plt.xlabel(attr) và plt.ylabel('Frequency')**: Gắn nhãn cho trục X với tên thuộc tính và trục Y với Frequency.
* **plt.grid(False)**: Tắt lưới trên biểu đồ.
* **safe\_attr = attr.replace("/", "\_")**: Thay thế ký tự / trong tên thuộc tính bằng \_ để tránh lỗi khi lưu tệp.
* **plt.savefig(f'{save\_path}/{team} {safe\_attr} Histogram.png')**: Lưu biểu đồ dưới dạng tệp ảnh PNG vào save\_path. Tên tệp bao gồm tên đội bóng (team), tên thuộc tính (safe\_attr) và "Histogram".
* **plt.show()**: Hiển thị biểu đồ.

**Kết quả: Tất cả histograms được lưu trong file Histograms.**

*e. Tìm đội bóng có chỉ số trung bình cao nhất ở mỗi chỉ số và đội có phong độ tốt nhất.*

**Bước 1: Đọc dữ liệu từ file CSV**

**Mã Python:**

df = pd.read\_csv('results2.csv')

**Giải thích:** Đọc dữ liệu từ tệp results2.csv và lưu nó vào DataFrame df. DataFrame này chứa thông tin các chỉ số trung vị, trung bình, độ lệch chuẩn về các đội bóng.

**Bước 2: Lọc các cột có chỉ số trung bình**

**Mã Python:**

mean\_columns = [col for col in df.columns if 'Mean of' in col]

**Giải thích:** Tạo một danh sách mean\_columns chứa các tên cột có chứa chuỗi ‘Mean of’ để lấy các chỉ số trung bình của các đội bóng nhằm thực hiện so sánh.

**Bước 3: Xác định đội bóng có chỉ số trung bình cao nhất cho từng cột**

**Mã Python:**

best\_teams = {}

cnt = {}

for col in mean\_columns:

    best\_team = df.loc[df[col].idxmax(), 'Team']

    best\_value = df[col].max()

    best\_teams[col] = (best\_team, best\_value)

    if best\_team in cnt:

        cnt[best\_team]+=1

    else: cnt[best\_team] = 1

**Giải thích:**

* **best\_teams**: Dùng để lưu đội bóng có chỉ số cao nhất ở mỗi chỉ số trung bình.
* **cnt**: Dùng để đếm số lần mỗi đội đứng đầu ở các chỉ số trung bình
* Vòng lặp for này duyệt qua từng cột trong mean\_columns.
* **best\_team = df.loc[df[col].idxmax(), 'Team']**: Tìm đội bóng có giá trị lớn nhất ở cột col bằng cách sử dụng idxmax() để tìm chỉ số của hàng có giá trị lớn nhất, sau đó lấy tên đội bóng trong cột 'Team'.
* **best\_value = df[col].max()**: Tìm giá trị lớn nhất của cột col.
* **best\_teams[col] = (best\_team, best\_value)**: Lưu đội bóng có chỉ số cao nhất và giá trị của chỉ số đó vào dictionary best\_teams.
* **Đếm số lần đội bóng đứng đầu**: Nếu best\_team đã có trong cnt, tăng giá trị của nó lên 1. Nếu không, khởi tạo cnt[best\_team] với giá trị 1.

**Bước 4: In kết quả đội bóng có chỉ số trung bình cao nhất ở mỗi chỉ số**

**Mã Python:**

print("Đội có chỉ số trung binh cao nhất cho từng chỉ số:")

for col, (team, value) in best\_teams.items():

    print(f"{col}: {team} với giá trị {value}")

**Giải thích:** Vòng lặp này duyệt qua best\_teams để in ra đội bóng có chỉ số trung bình cao nhất cho từng chỉ số và giá trị của nó.

**Bước 5: Xác định đội bóng có phong độ tốt nhất**

**Mã Python:**

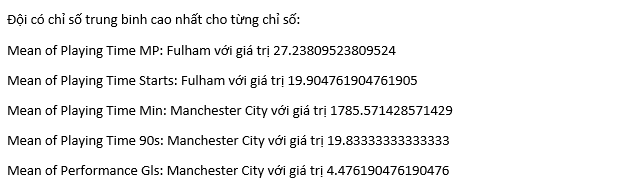
list = sorted(cnt, key=lambda i: cnt[i])

print("Đội có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024: " + list[len(list) - 1])

**Giải thích:**

* **list = sorted(cnt, key=lambda i: cnt[i])**: Sắp xếp các đội trong cnt theo số lần đứng đầu từ thấp đến cao.
* **list[len(list) - 1]**: Lấy đội cuối cùng trong danh sách list đã được sắp xếp, tức là đội có số lần đứng đầu nhiều nhất, và in ra đội có phong độ tốt nhất trong mùa giải 2023-2024.

**Kết quả mẫu (chi tiết kết quả lưu ở file highest\_index\_team):**

****

****

**III. (3đ)**

- Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau.

- Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? Bạn có Nhận xét gì về kết quả.

- Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.

- Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ với đầu vào như sau:

+ python radarChartPlot.py --p1 <player Name 1> --p2 <player Name 2> --Attribute <att1,att2,...,att\_n>

+ --p1: là tên cầu thủ thứ nhất + --p2: là tên cầu thủ thứ hai

+ --Attribute: là danh sách các chỉ số cần so sánh

+ Radar chart: https://matplotlib.org/stable/gallery/specialty\_plots/radar\_chart.html

**Bài làm:**

#### 1. Mục tiêu

1. Phân loại các cầu thủ thành các nhóm dựa trên chỉ số bằng thuật toán K-means.
2. Sử dụng thuật toán PCA để giảm số chiều dữ liệu xuống 2 chiều và trực quan hóa các nhóm bằng biểu đồ phân tán 2D.
3. Viết chương trình vẽ biểu đồ radar so sánh các chỉ số giữa hai cầu thủ theo các chỉ số được chỉ định.

#### 2. Các thư viện được sử dụng

* **Pandas**: Thư viện hỗ trợ thao tác và phân tích dữ liệu.
* **Matplotlib**: Thư viện hỗ trợ vẽ biểu đồ và trực quan hóa dữ liệu.
* **Scikit­-learn:** Thư viện phổ biến cho machine learning và phân tích dữ liệu trong Python. Nó bao gồm các thuật toán và công cụ cho phân loại, hồi quy, phân cụm, tiền xử lý dữ liệu và đánh giá mô hình.
* **Seaborn:** thư viện cho việc trực quan hóa dữ liệu, được xây dựng trên nền Matplotlib nhưng cung cấp các giao diện thân thiện hơn, màu sắc mặc định bắt mắt và phong phú.

#### 3. Các bước thực hiện và giải thích

#### *a. Phân cụm cầu thủ bằng thuật toán K-means.*

* **Mục đích**: Thuật toán K-means phân các cầu thủ thành các nhóm sao cho các cầu thủ trong cùng một nhóm có chỉ số tương tự nhau.
* **Chuẩn hóa dữ liệu**: StandardScaler được sử dụng để chuẩn hóa dữ liệu, giúp các đặc trưng có cùng tỷ lệ và ảnh hưởng công bằng đến kết quả phân cụm.
* **Phương pháp Elbow**: Để xác định số lượng cụm hợp lý, ta chạy K-means với các giá trị từ 2 đến 10 và ghi nhận giá trị “Inertia” (tổng bình phương khoảng cách từ điểm dữ liệu đến tâm cụm). Số cụm lý tưởng là khi đồ thị Inertia xuất hiện một “khuỷu tay” (elbow), tức là sau số cụm đó, việc tăng số lượng cụm không làm giảm đáng kể Inertia nữa.

#### *b. Giảm số chiều bằng PCA và vẽ biểu đồ phân cụm.*

* **Mục đích**: PCA (Principal Component Analysis) được sử dụng để giảm số chiều của dữ liệu xuống 2 chiều mà vẫn giữ lại được phần lớn thông tin, giúp trực quan hóa dễ dàng.
* **Thực hiện PCA**: PCA được thực hiện trên dữ liệu đã chuẩn hóa, tạo ra 2 thành phần chính (PC1 và PC2).
* **Phân cụm lại trên dữ liệu 2D**: Thuật toán K-means được áp dụng trên dữ liệu sau khi đã giảm số chiều xuống 2 để gán nhãn cụm cho các cầu thủ.
* **Trực quan hóa phân cụm**: Dữ liệu sau khi PCA và phân cụm được vẽ trên biểu đồ phân tán với màu sắc đại diện cho các cụm khác nhau, giúp dễ dàng nhận biết các nhóm cầu thủ có đặc điểm tương tự.

#### *c. Vẽ biểu đồ radar để so sánh cầu thủ.*

* **Mục tiêu**: Biểu đồ radar cho phép so sánh trực quan các chỉ số giữa hai cầu thủ được chỉ định. Mỗi trục trên biểu đồ radar đại diện cho một chỉ số được lựa chọn.
* **Cấu trúc hàm radar\_chart**:
  + **angles**: Được tính toán để tạo ra các góc cho các trục của biểu đồ radar.
  + **Dữ liệu cầu thủ**: Dữ liệu chỉ số của hai cầu thủ được đưa vào và thêm phần tử đầu vào cuối danh sách để đảm bảo vẽ được hình kín.
  + **Vẽ radar**: Biểu đồ radar được vẽ với hai màu khác nhau cho mỗi cầu thủ. Các điểm và đường nối được hiển thị với màu đỏ cho cầu thủ thứ nhất và màu xanh cho cầu thủ thứ hai. Các vùng màu dưới đường biểu diễn chỉ số của từng cầu thủ cũng được tô màu nhạt hơn để tăng tính trực quan.
  + **Kết quả**: Biểu đồ hiển thị cho phép so sánh nhanh các điểm mạnh và yếu của hai cầu thủ trên các chỉ số đã chọn.

#### 4. Phân tích chi tiết từng đoạn mã

*a.* *Phân cụm dữ liệu của các cầu thủ bằng thuật toán K-Means.*

**Bước 1: Đọc dữ liệu và xử lí các giá trị thiếu của cột chỉ số**

**Mã Python:**

df = pd.read\_csv('result.csv')

attributes = df.loc[:, df.columns[7:]]

attributes = attributes.map(lambda x: pd.to\_numeric(x, errors='coerce')).fillna(0)

**Giải thích:**

* Đọc dữ liệu từ file result.csv vào DataFrame df.
* Lựa chọn các cột bắt đầu từ cột thứ 8 (dùng df.columns[7:]) làm dữ liệu đặc trưng cho quá trình phân cụm.
* Áp dụng hàm pd.to\_numeric() cho tất cả các cột trong features để chuyển đổi chúng thành kiểu số, các giá trị không thể chuyển đổi sẽ được thay bằng NaN.
* Dùng fillna(0) để thay các giá trị NaN (thiếu) bằng 0.

**Bước 2: Chuẩn hóa dữ liệu**

**Mã Python:**

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(features)

**Giải thích:**

* Khởi tạo StandardScaler để chuẩn hóa dữ liệu.
* Chuẩn hóa dữ liệu trong features để các đặc trưng có giá trị trung bình 0 và độ lệch chuẩn 1, giúp K-Means hoạt động hiệu quả hơn.

**Bước 3: Xác định số cụm tối ưu bằng phương pháp Elbow**

**Mã Python:**

inertia = []

for k in range(2, 11):

    kmeans = KMeans(n\_clusters=k, random\_state=42)

    kmeans.fit(scaled\_features)

    inertia.append(kmeans.inertia\_)

**Giải thích:**

* Khởi tạo một danh sách inertia để lưu trữ giá trị của tổng bình phương khoảng cách từ các điểm đến tâm cụm.
* Chạy K-Means với số cụm từ 2 đến 10 và lưu giá trị inertia cho mỗi số cụm.

**Bước 4: Vẽ đồ thị phương pháp Elbow để xác định số cụm tối ưu**

**Mã Python:**

plt.plot(range(2, 11), inertia, marker='o')

plt.title('Phương pháp Elbow')

plt.xlabel('Số cụm')

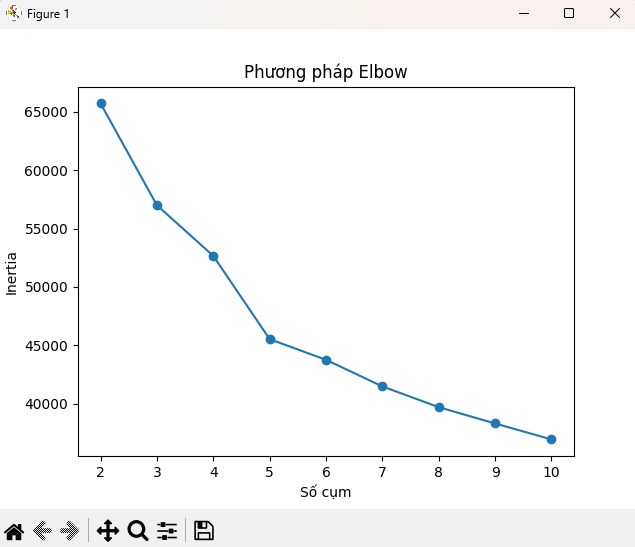
plt.ylabel('Inertia')

plt.show()

**Giải thích:**

* **plt.plot(range(2, 11), inertia, marker='o'):** Vẽ một đồ thị đường. Trục x biểu diễn số lượng cụm (từ 2 đến 10), trục y biểu diễn giá trị inertia. Các điểm dữ liệu trên đồ thị được đánh dấu bằng các vòng tròn.
* **plt.title('Phương pháp Elbow'), plt.xlabel('Số cụm'), plt.ylabel('Inertia'):** Đặt tiêu đề và nhãn cho các trục của đồ thị.
* **plt.show():** Hiển thị đồ thị

**Kết quả:**

****

**Bước 7: Kết luận về số cụm**

**Kết quả:** Dựa vào đồ thị được vẽ bới phương pháp Elbow ta có thể xác định được điểm gãy (khuỷu tay) của đồ thị nằm ở vị trị số cụm 5. Vậy phân nhóm cầu thủ thành 5 cụm là chính xác nhất.

**Nhận xét:** Dữ liệu có vẻ không dễ phân cụm rõ ràng, có thể do các cầu thủ trong dữ liệu không có sự khác biệt lớn về đặc trưng hoặc đặc trưng được sử dụng không phù hợp để phân biệt họ một cách rõ ràng.

*b. Giảm số chiều bằng PCA, vẽ hình phân cụm.*

**Bước 1: Giảm chiều dữ liệu bằng PCA**

**Mã Python:**

pca = PCA(n\_components=2)

pca\_result = pca.fit\_transform(scaled\_features)

**Giải thích:**

* **PCA (Principal Component Analysis):** là một kỹ thuật giảm chiều dữ liệu. PCA giúp giảm số chiều của dữ liệu xuống mà vẫn giữ lại càng nhiều thông tin quan trọng nhất có thể. Ở đây, dữ liệu gốc có nhiều đặc trưng (features), nhưng chúng ta chỉ giảm xuống còn 2 chiều (n\_components=2).
* **pca.fit\_transform(scaled\_features):** áp dụng PCA trên scaled\_features (dữ liệu đã được chuẩn hóa), trả về một mảng pca\_result với hai cột (2 chiều).

**Bước 2: Chuyển đổi kết quả PCA thành DataFrame và phân cụm bằng K-means**

**Mã Python:**

pca\_df = pd.DataFrame(pca\_result, columns=[‘PC1’, ‘PC2’])

kmeans = Kmeans(n\_clusters = 5)

kmeans.fit(pca\_result)

pca\_df[‘Cluster’] = kmeans.labels\_

**Giải thích:**

* Tạo một DataFrame mới từ pca\_result, với hai cột là PC1 và PC2
* Mỗi hàng trong pca\_df tương ứng với một cầu thủ và thể hiện tọa độ của cầu thủ đó trong không gian 2D mới sau khi áp dụng PCA.
* **kmeans.fit(pca\_result)**: huấn luyện mô hình K-means trên dữ liệu đã giảm chiều (tức là pca\_result).
* kmeans.labels\_ trả về nhãn của từng điểm dữ liệu (tức là cụm mà từng cầu thủ được gán vào). Nhãn này được thêm vào pca\_df dưới cột ‘Cluster’.

**Bước 3: Trực quan hoá kết quả phân cụm**

**Mã Python:**

plt.figure(figsize=(8, 6))

sns.scatterplot(x=’PC1’, y=’PC2’, hue=’Cluster’, data=pca\_df, palette=’viridis’)

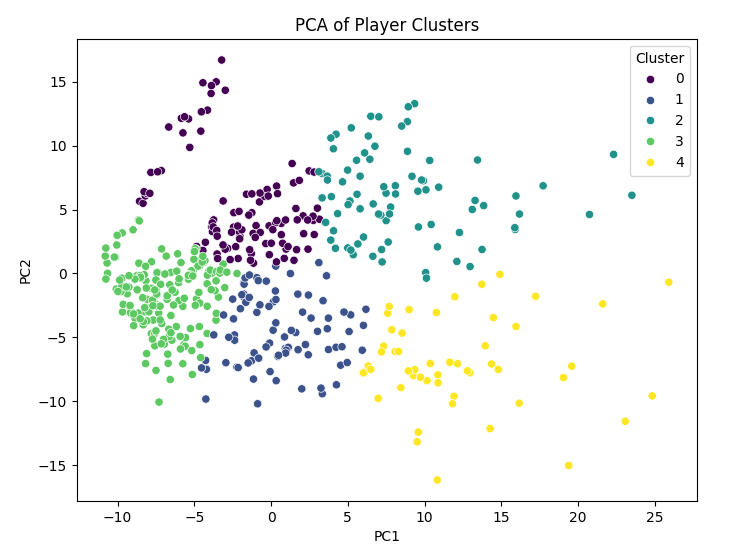
plt.title(‘PCA of Player Clusters’)

plt.show()

**Giải thích:**

* **Trực quan hóa phân cụm trên mặt phẳng 2D**: Sử dụng Seaborn để vẽ biểu đồ phân tán với trục x là PC1, trục y là PC2.
* **Màu sắc của các cụm**: hue=’Cluster’ dùng để phân biệt các cụm bằng màu sắc khác nhau, với bảng màu là ‘viridis’.

**Kết quả:**

****

*c. Chương trình Python vẽ biểu đồ rada so sánh cầu thủ.*

**Bước 1: Định nghĩa hàm radar\_chart**

**Mã Python:**

def radar\_chart(player1\_data, player2\_data, attributes, player1\_name, player2\_name):

    num\_vars = len(attributes)

...

    plt.show()

**Giải thích:**

* Hàm radar\_chart được định nghĩa để vẽ biểu đồ radar cho hai cầu thủ dựa trên dữ liệu chỉ số của họ (player1\_data và player2\_data).
* Các tham số của hàm:
* player1\_data, player2\_data: Danh sách chứa giá trị của các thuộc tính (chỉ số) của cầu thủ thứ nhất và thứ hai.
* attributes: Danh sách tên các chỉ số cần so sánh.
* player1\_name, player2\_name: Tên của cầu thủ thứ nhất và thứ hai để hiển thị trên biểu đồ.

**Bước 2: Thiết lập góc cho các chỉ số**

**Mã Python:**

num\_vars = len(attributes)

angles = np.linspace(0, 2 \* np.pi, num\_vars, endpoint=False).tolist()

**Giải thích:**

* **num\_var**: danh sách các thuộc tính cần so sánh.
* **angles**: danh sách các góc trên biểu đồ radar, với mỗi chỉ số (attribute) có một góc tương ứng.
* **np.linspace(0, 2 \* np.pi, num\_vars, endpoint=False**): tạo một mảng các giá trị từ 0 đến 2π, chia đều theo số lượng chỉ số (num\_vars).

**Bước 3: Đóng vòng dữ liệu cho biểu đồ radar**

**Mã Python:**

player1\_data += player1\_data[:1]

player2\_data += player2\_data[:1]

angles += angles[:1]

**Giải thích:**

* Giá trị đầu tiên của player1\_data và player2\_data được thêm vào cuối mỗi danh sách. Điều này tạo một vòng tròn khép kín khi vẽ biểu đồ.
* **angles** cũng được thêm góc đầu tiên vào cuối để khép kín vòng các góc.

**Bước 4: Tạo biểu đồ, vẽ vùng và đường trên biểu đồ radar**

**Mã Python:**

fig, ax = plt.subplots(figsize=(6, 6), subplot\_kw=dict(polar=True))

ax.fill(angles, player1\_data, color='red', alpha=0.25)

ax.fill(angles, player2\_data, color='blue', alpha=0.25)

ax.plot(angles, player1\_data, color='red', linewidth=2, label=player1\_name)

ax.plot(angles, player2\_data, color='blue', linewidth=2, label=player2\_name)

**Giải thích:**

* Thiết lập polar=True trong subplot\_kw biến hệ trục ax thành trục tọa độ cực (polar coordinates), cần thiết để tạo biểu đồ radar.
* **ax.fill** tạo các vùng màu sắc nhạt (alpha = 0.25) cho hai cầu thủ, giúp dễ dàng so sánh mức độ giữa các chỉ số.
* **ax.plot** vẽ các đường bao quanh từng khu vực của hai cầu thủ, với màu đỏ và xanh để phân biệt.

**Bước 5:** **Thiết lập nhãn, tiêu đề, chú thích và hiển thị biểu đồ**

**Mã Python:**

ax.set\_yticklabels([])

ax.set\_xticks(angles[:-1])

ax.set\_xticklabels(attributes)

plt.legend(loc='upper right')

plt.title(f'{player1\_name} vs {player2\_name} Comparison')

plt.show()

**Giải thích:**

* **ax.set\_yticklabels([]):** ẩn các nhãn trên trục y (khoảng cách từ tâm đến các điểm trên biểu đồ radar).
* **ax.set\_xticks(angles[:-1]) và ax.set\_xticklabels(attributes):** thiết lập các nhãn cho mỗi góc trên biểu đồ radar, sử dụng tên chỉ số trong attributes.
* **plt.legend(loc='upper right'):** Thêm chú thích ở góc trên bên phải để phân biệt hai cầu thủ.
* **plt.title:** Thêm tiêu đề biểu đồ so sánh hai cầu thủ dựa trên các chỉ số.

**Bước 6:** **Lấy dữ liệu từ bảng result.csv và** **nhập dữ liệu cho cầu thủ và các chỉ số**

**Mã Python:**

df = pd.read\_csv('result.csv')

player1\_name = input("Nhập tên cầu thủ thứ nhất: ")

player2\_name = input("Nhập tên cầu thủ thứ hai: ")

attributes = []

while True:

    attribute = input("Nhập tên thuộc tính (hoặc 'xong' để kết thúc): ")

    if attribute.lower() == 'xong':

        break

    attributes.append(attribute)

**Giải thích:**

* Đọc dữ liệu từ file result.csv.
* Nhập tên của hai cầu thủ cần so sánh và các chỉ số cần dùng trên biểu đồ radar.

**Bước 7:** **Trích xuất dữ liệu cho hai cầu thủ**

**Mã Python:**

player1\_data = df[df['Player'] == player1\_name] [attributes].values.flatten().tolist()

player2\_data = df[df['Player'] == player2\_name] [attributes].values.flatten().tolist()

**Giải thích:**

* **df[df['Player'] == player1\_name]** và **df[df['Player'] == player2\_name]**:
* df là một DataFrame chứa dữ liệu về tất cả các cầu thủ.
* df['Player'] == player1\_name tạo một điều kiện để lọc DataFrame, giữ lại các hàng mà cột 'Player' có giá trị bằng player1\_name (tên của cầu thủ thứ nhất).
* Tương tự, df['Player'] == player2\_name lọc ra các hàng mà tên cầu thủ là player2\_name (tên của cầu thủ thứ hai).
* Sau khi lọc ra đúng cầu thủ, [attributes] lấy các cột tương ứng với danh sách thuộc tính đã chọn. attributes là danh sách các chỉ số mà người dùng muốn so sánh (như "Playing Time MP", " Shooting Standard Gls ", v.v.).
* **.values** chuyển các giá trị trong DataFrame được lọc thành một mảng NumPy, để dễ dàng xử lý và truy cập dữ liệu.
* **.flatten()** chuyển đổi mảng 2 chiều (có thể có một hàng và nhiều cột) thành một mảng 1 chiều, giúp việc xử lý các giá trị dễ dàng hơn.
* **.tolist()** chuyển mảng NumPy thành một danh sách Python để dễ thao tác hơn sau này.

**Bước 8:** **Gọi hàm radar\_chart để vẽ biểu đồ**

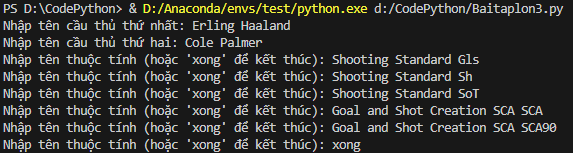
**Mã Python:**

radar\_chart(player1\_data, player2\_data, attributes, player1\_name, player2\_name)

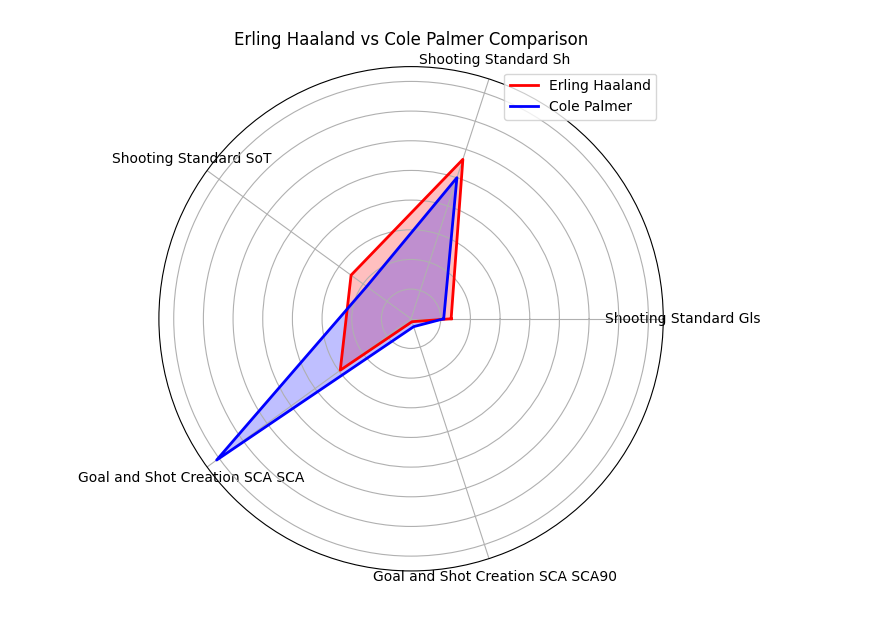
**Giải thích:** Gọi hàm radar\_chart với dữ liệu của hai cầu thủ để vẽ biểu đồ radar.

**Kết quả mẫu:** So sánh 2 cầu thủ Erling Haaland và Cole Palmer dựa trên các chỉ số: Goals, Shots Total, Shoot on Target, Shot Creating Actions, Shot Creating Actions/90

* Input:



* Output:



**IV. (1đ)**

- Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024 từ trang web <https://www.footballtransfers.com>.

- Đề xuất phương pháp định giá cầu thủ.

**Bài làm:**

### 1. Mục tiêu

Mục tiêu của dự án là thu thập và phân tích dữ liệu về giá chuyển nhượng cầu thủ trong mùa giải 2023-2024 từ trang web FootballTransfers, từ đó phát triển một phương pháp định giá cầu thủ dựa trên các chỉ số thống kê. Cụ thể, các mục tiêu bao gồm:

* Thu thập dữ liệu giá chuyển nhượng của các cầu thủ.
* Xử lý và chuyển đổi dữ liệu để phù hợp với định dạng cần thiết.
* Phát triển mô hình hồi quy tuyến tính để dự đoán giá trị chuyển nhượng dựa trên các chỉ số cầu thủ.
* Cung cấp khả năng dự đoán giá trị chuyển nhượng cho người dùng.

### 2. Công cụ và Thư viện Sử dụng

* **Python**: Ngôn ngữ lập trình chính được sử dụng trong dự án.
* **Selenium**: Thư viện để tự động hóa việc duyệt web và thu thập dữ liệu từ trang web.
* **BeautifulSoup**: Thư viện để phân tích cú pháp HTML và trích xuất thông tin từ các trang web.
* **Pandas**: Thư viện để xử lý và phân tích dữ liệu, đặc biệt là để tạo và quản lý các DataFrame.
* **Scikit-learn**: Thư viện cho học máy, được sử dụng để xây dựng và đánh giá mô hình hồi quy tuyến tính.

### 3. Các phương pháp giải bài toán

*a, Thu thập giá chuyển nhượng cầu thủ*

Dữ liệu về giá chuyển nhượng được thu thập từ trang web FootballTransfers thông qua quy trình tự động hóa:

* **Sử dụng Selenium**: Đầu tiên, Selenium WebDriver được sử dụng để điều khiển trình duyệt web tự động. Điều này cho phép truy cập vào trang web và thực hiện việc tải các trang cần thiết để thu thập thông tin.
* **Lấy dữ liệu từ HTML**: Sau khi trang web được tải, BeautifulSoup được sử dụng để phân tích cú pháp HTML. Nó giúp trích xuất thông tin từ bảng chứa dữ liệu về các cầu thủ và giá chuyển nhượng. Các thông tin được thu thập bao gồm:
  + Tên cầu thủ
  + Giá chuyển nhượng
* **Lưu trữ dữ liệu**: Dữ liệu thu thập được được lưu vào một DataFrame của Pandas và sau đó xuất ra dưới dạng tệp CSV để phục vụ cho các bước xử lý và phân tích sau này.

*b, Phương pháp định giá cầu thủ*

Sau khi thu thập dữ liệu, dự án sử dụng mô hình hồi quy tuyến tính để định giá cầu thủ dựa trên các chỉ số thống kê:

* **Chuẩn bị dữ liệu**: Các dữ liệu giá chuyển nhượng được chuyển đổi thành định dạng số (triệu Euro), và dữ liệu thống kê cầu thủ được kết hợp với giá chuyển nhượng bằng cách gộp các DataFrame lại với nhau.
* **Xử lý và phân loại**: Dữ liệu được tiền xử lý để đảm bảo tính đồng nhất. Các biến đặc trưng (features) được tách ra và chuẩn hóa, trong khi giá trị chuyển nhượng được xác định là biến mục tiêu (target) cho mô hình.
* **Xây dựng mô hình**: Mô hình hồi quy tuyến tính được sử dụng để học từ dữ liệu huấn luyện. Mô hình này sẽ tìm ra mối quan hệ giữa các chỉ số cầu thủ và giá trị chuyển nhượng của họ.
* **Dự đoán và đánh giá**: Người dùng có thể nhập tên cầu thủ để nhận dự đoán về giá trị chuyển nhượng của cầu thủ đó dựa trên các chỉ số của họ.

#### 4. Phân tích chi tiết từng đoạn mã

*Bước 1: Thu thập dữ liệu*

**Mã Python:**

base\_url = "https://www.footballtransfers.com/us/transfers/confirmed/2023-2024/uk-premier-league/fee-to-200000000/"

transfers = []

headers = ["Player", "Price"]

for page\_num in range(1, 9):

    driver = webdriver.Chrome()

    driver.get(base\_url + str(page\_num))

    driver.implicitly\_wait(10)

    html = driver.page\_source

    soup = BeautifulSoup(html, 'html.parser')

    table = soup.find("table", {'class':'table table-striped table-hover leaguetable mvp-table transfer-table mb-0'})

    for row in table.find('tbody').find\_all('tr'):

        player\_name = row.find\_all('td')[0].find('span').get\_text()

        player\_price = row.find\_all('td')[3].find('span').get\_text()

        transfers.append([player\_name, player\_price])

dfstats = pd.DataFrame(transfers, columns=headers)

dfstats.to\_csv('results4.csv')

**Giải thích:** Khởi tạo Selenium WebDriver để truy cập vào trang web và lặp qua các trang để thu thập tên cầu thủ và giá chuyển nhượng. Dữ liệu sau đó được lưu vào DataFrame và xuất ra tệp “result4.csv”.

*Bước 2: Xử lí dữ liệu*

**Mã Python:**

df\_transfer = pd.read\_csv('results4.csv')

df\_stats = pd.read\_csv('result.csv')

def convert\_to\_million(value):

    if isinstance(value, str) and value == "Free":

        return 0

    if isinstance(value, str) and value.endswith('K'):

        return float(value[1:-1]) / 1000

    else:

        return float(value[1:-1])

df\_transfer['Price'] = df\_transfer['Price'].apply(convert\_to\_million)

**Giải thích:**

* Đọc dữ liệu từ file chứa tên cầu thủ và giá trị chuyển nhượng(‘result4.csv’) và file chứa tên cầu thủ và các chỉ số(‘result.csv’).
* Chuyển các giá trị ở trong bảng cột ‘Price’ về dạng số thực:
  + Nếu là “Free” thì trả về giá trị 0.
  + Nếu có đuôi là “K” thì trả về giá trị thực/1000.
  + Nếu có đuôi là giá trị khác(“M”) thì trả về giả trị thực.

*Bước 3: Gộp dữ liệu và tiền xử lý*

**Mã Python:**

matched\_data = df\_transfer.merge(df\_stats, on='Player', how='inner')

matched\_data = matched\_data.drop(columns=['Unnamed: 0\_x', 'Unnamed: 0\_y', 'Nation', 'Pos', 'Squad', 'Age', 'Born'])

features = matched\_data.loc[:, matched\_data.columns[2:]]

features = features.apply(pd.to\_numeric, errors='coerce').fillna(0)

target = matched\_data['Price']

**Giải thích:**

* Gộp dữ liệu từ 2 bảng dựa trên tiêu chí là tên cầu thủ và loại bỏ các cột không cần thiết cho việc phân tích.
* Tạo ra biến “features” lưu các đặc trưng cho các cầu thủ dùng cho việc phân tích cùng và xử lý dữ liệu thành dạng số thực.
* Tạo ra biến “target” làm giá trị mà mô hình cần dự đoán. Trong trường hợp này, biến “target” là giá chuyển nhượng của các cầu thủ

*Bước 4: Huấn luyện mô hình*

**Mã Python:**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, random\_state=42)

model = LinearRegression()

model.fit(X\_train, y\_train)

**Giải thích:**

* **train\_test\_split**: Đây là một hàm từ thư viện sklearn.model\_selection, được sử dụng để chia tập dữ liệu thành hai phần: tập huấn luyện và tập kiểm tra.
* **features**: Là biến chứa các đặc trưng (các chỉ số cầu thủ) mà mô hình sẽ học từ đó. **target**: Là biến mục tiêu (giá trị chuyển nhượng của cầu thủ) mà mô hình sẽ cố gắng dự đoán.
* **test\_size=0.2**: Tham số này chỉ định tỷ lệ dữ liệu sẽ được sử dụng cho tập kiểm tra. Trong trường hợp này, 20% dữ liệu sẽ được sử dụng để kiểm tra mô hình sau khi đã huấn luyện.
* **random\_state=42**: Đây là một số hạt giống ngẫu nhiên để đảm bảo rằng mỗi lần chạy mã đều có kết quả tương tự. Điều này hữu ích cho việc tái tạo kết quả.
* **model = LinearRegression():** Khởi tạo mô hình hồi quy tuyến tính. Khi khởi tạo mô hình này, nó sẽ được sử dụng để dự đoán giá trị mục tiêu dựa trên các đặc trưng đầu vào.
* **model.fit(X\_train, y\_train)**: Hàm này sẽ huấn luyện mô hình hồi quy tuyến tính bằng cách sử dụng dữ liệu trong X\_train (các đặc trưng) và y\_train (biến mục tiêu). Mô hình sẽ tìm ra các trọng số tối ưu cho các đặc trưng sao cho dự đoán của nó gần nhất với giá trị thực tế trong y\_train.

*Bước 5: Dự đoán giá trị chuyển nhượng*

**Mã Python:**

player\_name = input("Nhập tên cầu thủ để dự đoán giá trị chuyển nhượng: ")

if player\_name in df\_stats['Player'].values:

    player\_stats = df\_stats[df\_stats['Player'] == player\_name].loc[:, matched\_data.columns[2:]]

    player\_stats = player\_stats.apply(pd.to\_numeric, errors='coerce').fillna(0)

    predicted\_price = model.predict(player\_stats)

    print(f"Giá trị chuyển nhượng dự đoán cho {player\_name}: €{round(predicted\_price[0])}M")

else:

    print("Không tìm thấy cầu thủ trong dữ liệu.")

**Giải thích:**

* **if player\_name in df\_stats['Player'].values**: Kiểm tra xem cầu thủ trong dữ liệu không
* **df\_stats[df\_stats['Player'] == player\_name]**: Lọc ra hàng trong DataFrame df\_stats tương ứng với cầu thủ có tên player\_name.
* **.loc[:, matched\_data.columns[2:]]**: Chọn tất cả các cột từ cột thứ 2 trở đi (các chỉ số cầu thủ) để tạo ra DataFrame player\_stats, chỉ giữ lại các đặc trưng mà mô hình đã học.
* **player\_stats = player\_stats.apply(pd.to\_numeric, errors='coerce').fillna(0**): xử lý dữ liệu thành dạng số thực
* **model.predict(player\_stats)**: Sử dụng mô hình đã huấn luyện để dự đoán giá trị chuyển nhượng dựa trên các chỉ số của cầu thủ trong player\_stats.
* **print()**: In ra giá trị chuyển nhượng dự đoán cho cầu thủ, sử dụng round(predicted\_price[0]) để làm tròn giá trị dự đoán xuống đơn vị triệu euro.
* **else: print("Không tìm thấy cầu thủ trong dữ liệu."):** Nếu tên cầu thủ không được tìm thấy trong dữ liệu, mã sẽ thông báo cho người dùng rằng cầu thủ đó không có trong cơ sở dữ liệu.

**Kết quả:**

