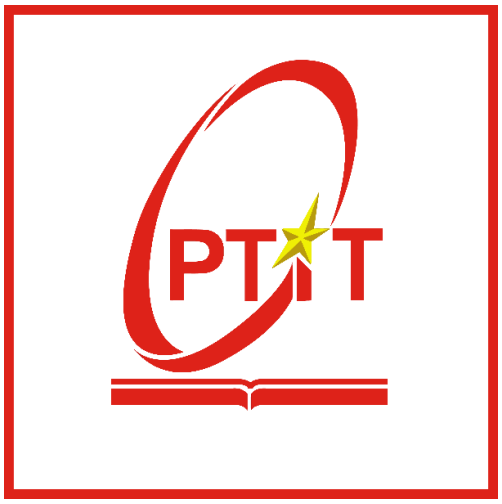
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1



BÁO CÁO BÀI TẬP
MÔN HỌC: LẬP TRÌNH VỚI PYTHON

Giảng viên	**:**	**Thầy Kim Ngọc Bách**
Sinh viên	**:**	**Nguyễn Hoài Nam**
Lớp	**:**	**D22CQCN08-B**
Mã sinh viên	**:**	**B22DCCN560**

Tháng 10/2024

Contents

Bài 1:

I. Phân tích yêu cầu bài toán :

1. Mục tiêu tổng quan của bài toán

Viết chương trình thu thập dữ liệu thống kê về các cầu thủ thi đấu tại giải Ngoại hạng Anh mùa 2023-2024 từ trang fbref.com, sau đó xử lý và ghi kết quả ra file results.csv. Mỗi cầu thủ phải có đủ thông tin về các chỉ số quy định, và chỉ những cầu thủ có số phút thi đấu lớn hơn 90 phút mới được tính. Dữ liệu sau khi thu thập được sắp xếp và xử lý theo thứ tự cụ thể để đảm bảo đúng định dạng yêu cầu.

2. Yêu cầu chi tiết và cấu trúc dữ liệu đầu ra

2.1 Nguồn dữ liệu

- **Website**: https://fbref.com/en/
- **Đối tượng thu thập**: Tất cả các cầu thủ có số phút thi đấu nhiều hơn 90 phút tại giải Ngoại hạng Anh mùa giải 2023-2024.
- **URL tham khảo cụ thể**: Một ví dụ về trang dữ liệu của một đội bóng là trang thống kê của Liverpool: https://fbref.com/en/squads/822bd0ba/2023-2024/Liverpool-Stats. Từ đây, có thể suy ra cách tổ chức các URL của các đội khác.

2.2 Các chỉ số cần thu thập

Mỗi cầu thủ phải có các chỉ số sau. Mỗi cột của bảng kết quả sẽ tương ứng với một chỉ số thống kê, và nếu chỉ số nào không có hoặc không áp dụng, thì sẽ thay bằng "N/a".

- **Thông tin cơ bản**:
  - Nation: Quốc gia
  - Team: Đội bóng
  - Position: Vị trí
  - Age: Tuổi
- **Thời gian thi đấu**:
  - Matches Played: Số trận đã thi đấu
  - Starts: Số lần ra sân từ đầu trận
  - Minutes: Số phút thi đấu
- **Hiệu suất thi đấu**:
  - Non-Penalty Goals: Số bàn thắng không bao gồm phạt đền
  - Penalty Goals: Số bàn thắng từ phạt đền
  - Assists: Kiến tạo
  - Yellow Cards: Số thẻ vàng
  - Red Cards: Số thẻ đỏ
- **Các chỉ số kỳ vọng**:
  - xG: Expected Goals (bàn thắng kỳ vọng)

- npxG: Non-Penalty Expected Goals (bàn thắng kỳ vọng không bao gồm phạt đền)
- xAG: Expected Assists (kiến tạo kỳ vọng)

➢ **Chỉ số tiến bộ (Progression)**:
- PrgC: Passes that progressed the ball towards the opponent's goal
- PrgP: Progressive Passes
- PrgR: Progressive Carries

➢ **Chỉ số mỗi 90 phút thi đấu**:
- Gls, Ast, G+A, G-PK, G+A-PK, xG, xAG, xG + xAG, npxG, npxG + xAG

➢ **Chỉ số cho thủ môn**:
- **Hiệu suất**:
  - GA: Goals Against
  - GA90: Goals Against per 90 minutes
  - SoTA: Shots on Target Against
  - Saves: Số lần cứu thua
  - Save%: Tỷ lệ cứu thua
  - W, D, L: Số trận thắng, hòa và thua
  - CS: Clean Sheets
  - CS%: Tỷ lệ Clean Sheets
- **Phạt đền**:
  - PKatt: Penalty Kicks Attempted
  - PKA: Penalty Kicks Allowed
  - PKsv: Penalty Kicks Saved
  - PKm: Penalty Kicks Missed
  - Save%: Tỷ lệ cứu thua phạt đền

➢ **Chỉ số sút bóng (Shooting)**:
- **Chuẩn**:
  - Gls: Goals
  - Sh: Shots
  - SoT: Shots on Target
  - SoT%: Tỷ lệ sút trúng đích
  - Sh/90: Số lần sút mỗi 90 phút
  - SoT/90: Số lần sút trúng đích mỗi 90 phút
  - G/Sh, G/SoT: Bàn thắng mỗi lần sút hoặc mỗi lần sút trúng đích
  - Dist: Khoảng cách sút trung bình
  - FK: Free Kicks
  - PK, PKatt: Số bàn thắng và số lần thực hiện phạt đền
- **Kỳ vọng**:
  - xG, npxG, npxG/Sh, G-xG, np:G-xG

➢ **Chuyền bóng (Passing)**:
- **Tổng thể**: Cmp, Att, Cmp%, TotDist, PrgDist
- **Ngắn (Short)**: Cmp, Att, Cmp%

- **Trung bình (Medium)**: Cmp, Att, Cmp%
- **Dài (Long)**: Cmp, Att, Cmp%
- **Kỳ vọng**:
  - Ast, xAG, xA, A-xAG, KP, 1/3, PPA, CrsPA, PrgP

➢ **Loại đường chuyền (Pass Types)**:
- Các loại: Live, Dead, FK, TB, Sw, Crs, TI, CK
- Phạt góc (Corner Kicks): In, Out, Str
- Kết quả: Cmp, Off, Blocks

➢ **Tạo cơ hội ghi bàn và sút bóng (Goal and Shot Creation)**:
- **SCA**: SCA, SCA90
- **Loại tạo cơ hội (SCA Types)**: PassLive, PassDead, TO, Sh, Fld, Def
- **GCA**: GCA, GCA90
- **Loại tạo cơ hội ghi bàn (GCA Types)**: PassLive, PassDead, TO, Sh, Fld, Def

➢ **Hành động phòng ngự (Defensive Actions)**:
- **Tackles**: Tkl, TklW, Def 3rd, Mid 3rd, Att 3rd
- **Challenges**: Tkl, Att, Tkl%, Lost
- **Blocks**: Blocks, Sh, Pass, Int, Tkl + Int, Clr, Err

➢ **Kiểm soát bóng (Possession)**:
- **Touches**: Touches, Def Pen, Def 3rd, Mid 3rd, Att 3rd, Att Pen, Live
- **Take-Ons**: Att, Succ, Succ%, Tkld, Tkld%
- **Carries**: Carries, TotDist, ProDist, ProgC, 1/3, CPA, Mis, Dis
- **Receiving**: Rec, PrgR

➢ **Thời gian thi đấu và thành công của đội (Playing Time & Team Success)**:
- **Bắt đầu (Starts)**: Starts, Mn/Start, Compl
- **Dự bị (Subs)**: Subs, Mn/Sub, unSub
- **Đội hình**: PPM, onG, onGA, onxG, onxGA

➢ **Các chỉ số khác (Miscellaneous Stats)**:
- **Hiệu suất**: Fls, Fld, Off, Crs, OG, Recov
- **Không chiến**: Won, Lost, Won%

3. Định dạng và sắp xếp dữ liệu

➢ Kết quả được ghi vào file results.csv.
➢ Sắp xếp cầu thủ theo tên (First Name). Nếu trùng tên, sắp xếp theo độ tuổi từ lớn đến nhỏ.
➢ Bất kỳ giá trị nào thiếu hoặc không áp dụng sẽ được điền là "N/a".

II. Giải quết vấn đề

1. Thuật toán

- Do cần thu thập dữ liệu từ nhiều đường dẫn khác nhau nên sẽ có 2 công việc chính:
    - 1 là lấy dữ liệu từ các đường dẫn
    - 2 là gộp các bảng lại và xóa các trường dữ liệu thừa
- Các thư viện được sử dụng trong bài tập này
    - Beautifulsoup
    - Requests
    - Pandas

1.1. Lấy dữ liệu từ các đường dẫn

- Các bước thực hiện
    - Gửi request tới dường dẫn để nhận file html về
    - Chuyển sang dạng tags html
    - Tìm kiếu thẻ div chứa nội dung cần lấy
    - Vì nội dung cần lấy khi kéo về ở trong comment nên cần thực hiện lấy ra phần comments rồi đưa lại về dạng tags(ví dụ em để trong file output.html)
    - Tên các cột lấy theo thuộc tính ‘data-stat’. Do giá trị của thuộc tính trung với tên cột
    - Lưu dữ liều vào dictionary rồi chuyển về dataframe
    - Lưu bảng thành các file .csv
- Do nhận thấy các bảng từ 2-> 10 đều lấy toàn bộ bảng và xóa các trường dữ liệu giống nhau nên em dặt vào vòng lặp tách riêng so với thực hiện thu thập dữ liệu từ bảng 1.

1.2. Gộp các bảng lại và xóa các trường dữ liệu không cần thiết

- Các bảng 1 và từ 3 đến 10 dều có cột ‘Unname :0’(cột đánh chỉ số các cầu thủ) giống nhau.
    - Ghép các bảng 1 và từ 3 đến 10 dựa vào cột ‘Unname :0’.(gọi bảng này là bảng result)
- Do tên các thủ môn không trùng nhau nên bảng 2 sẽ ghép với bảng result đựa trên tên cầu thủ.
- Tiếp sẽ đưa chỉ số minutes về dạng số và xóa các cầu thủ có thời gian thi đầu <= 90 phút
- Sắp xếp lại bảng theo tên cầu thủ tăng dần và tuổi giảm dần
- Lưu bàng vào file result.csv

2. Phân tích code

❖ Hàm cr1(url, id_div, cnt)

- Đây là hàm thu thập dữ liệu từ một URL chỉ định và lưu dữ liệu thành tệp CSV.
- Quy trình hoạt động của cr1:
  - Bước 1: Tải nội dung HTML từ URL thông qua "requests.get(url)".
  - Bước 2: Phân tích HTML bằng BeautifulSoup để tìm phần div chứa dữ liệu dựa trên "id_div".
  - Bước 3: Tìm các bình luận HTML trong div, vì dữ liệu cần lấy nằm trong các bình luận.
  - Bước 4: Lọc các thẻ th trong dòng đầu của tr để lấy tên các cột, lưu vào "key1" (bỏ qua th đầu tiên).
  - Bước 5: Lặp qua các dòng tr còn lại để lấy dữ liệu của từng cầu thủ và lưu vào "key1".
    - Xử lý đặc biệt: Nếu "data-stat" là nationality, chỉ lấy mã quốc gia (chỉ lấy phần đầu khi tách chuỗi bằng dấu cách).
  - Bước 6: Tạo DataFrame từ "key1" và lưu vào tệp table{cnt}.csv.
    - Xử lý đặc biệt: Nếu cnt == 2, loại bỏ các cột gk_games, gk_games_starts, gk_minutes, và minutes_90s.

❖ Hàm clean_data()

- Hàm này xử lý và hợp nhất dữ liệu từ các tệp CSV đã tạo, sau đó lưu dữ liệu đã xử lý vào "result.csv".
- Các bước chính của clean_data():
  - Bước 1: Đọc tệp "bang1.csv" làm dữ liệu ban đầu.
  - Bước 2: Với mỗi tệp từ "table3.csv" đến "table10.csv", chỉ giữ lại các cột không trùng với "result" (ngoại trừ cột Unnamed: 0) và hợp nhất vào result theo phương thức inner join.
  - Bước 3: Xác định các cột chung giữa "result" và "table2.csv" để hợp nhất hai bảng này.
  - Bước 4: Loại bỏ các cột không cần thiết (Unnamed: 0_x, Unnamed: 0_y, minutes_90s, birth_year, matches).
  - Bước 5: Chuyển cột minutes thành kiểu số nguyên, loại bỏ mọi dấu phẩy trong chuỗi số.
  - Bước 6: Chỉ giữ lại các cầu thủ có minutes > 90.
  - Bước 7: Sắp xếp dữ liệu theo player (tên cầu thủ) và age (tuổi), rồi lưu vào "result.csv".

❖ Khối if __name__ == '__main__':

- Đây là phần chính để thực thi mã, bao gồm:
- Tải trang chính và phân tích lấy all_stats_standard.
- Tạo DataFrame đầu tiên từ dữ liệu lấy được và lưu vào "bang1.csv".
- Định nghĩa các URL và id của từng bảng con cần lấy dữ liệu, như keepers, shooting, passing, v.v.

- Vòng lặp gọi hàm cr1 cho từng URL, lưu mỗi bảng vào các tệp CSV tương ứng từ “table2.csv” đến “table10.csv.”
- Gọi clean_data() để hợp nhất và làm sạch dữ liệu cuối cùng.

3. Kết quả:

[3]:

	Unnamed: 0	player	nationality	position	team	age	games	games_starts	minutes	assists	...	gk_wins	gk_ties	gk_losses	gk_clean_sheets	gk_clean_s
0	0	Max Aarons	eng	DF	Bournemouth	23	20	13	1237	1	...	NaN	NaN	NaN	NaN	
1	2	Tyler Adams	us	MF	Bournemouth	24	3	1	121	0	...	NaN	NaN	NaN	NaN	
2	3	Tosin Adarabioyo	eng	DF	Fulham	25	20	18	1617	0	...	NaN	NaN	NaN	NaN	
3	4	Elijah Adebayo	eng	FW	Luton Town	25	27	16	1419	0	...	NaN	NaN	NaN	NaN	
4	5	Simon Adingra	ci	FW	Brighton	21	31	25	2222	1	...	NaN	NaN	NaN	NaN	
5	6	Nayef Aguerd	ma	DF	West Ham	27	21	21	1857	0	...	NaN	NaN	NaN	NaN	
6	8	Naouirou Ahamada	fr	MF,FW	Crystal Palace	21	20	0	349	0	...	NaN	NaN	NaN	NaN	
7	9	Anel Ahmedhodžić	ba	DF	Sheffield Utd	24	31	29	2649	0	...	NaN	NaN	NaN	NaN	
8	10	Ola Aina	ng	DF	Nott'ham Forest	26	22	20	1692	1	...	NaN	NaN	NaN	NaN	
9	11	Rayan Aït-Nouri	dz	DF,MF	Wolves	22	33	29	2329	1	...	NaN	NaN	NaN	NaN	

Bài 2:

I. Phân tích yêu cầu bài toáns :

Yêu cầu này yêu cầu chúng ta thực hiện các phép tính thống kê và trực quan hóa dữ liệu trên tập dữ liệu cầu thủ bóng đá đã thu thập được. Cụ thể:

- Tìm top 3: Xác định 3 cầu thủ có giá trị cao nhất và thấp nhất cho mỗi chỉ số.
- Tính toán các thống kê: Tính trung vị, trung bình, và độ lệch chuẩn cho mỗi chỉ số trên toàn bộ dữ liệu và cho từng đội.
- Tạo file CSV: Lưu kết quả các phép tính thống kê vào file CSV theo định dạng cho trước.
- Vẽ biểu đồ: Vẽ biểu đồ histogram để trực quan hóa phân bố của mỗi chỉ số.
- Phân tích: Dựa trên các kết quả, đưa ra nhận xét về đội có chỉ số cao nhất và đội có phong độ tốt nhất.

II. Giải quết vấn đề

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

- Code :

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

[3]:
```
for x in df.columns[5:]:
    df_copy =df.copy()
    if 'gk' in x:
        df_copy= df_copy[df_copy['position']=='GK']
    tmp =df_copy[['player',x]].sort_values(x)
    tmp.dropna(axis=0,inplace=True)
    print(f'3 cầu thủ có {x} cao nhất')
    display(tmp[-3:][::-1])
    print(f'3 cầu thủ có {x} thấp nhất')
    display(tmp[:3])
```

- Mô tả:
  - Để tránh làm sai dữ liệu đã thu thập được nên em sẽ tạo một bản copy
  - Đối với những chỉ số của riêng thủ môn thì em sẽ chỉ so sánh những thủ môn với nhau
  - Lấy ra từ bảng 2 trường thông tin là tên cầu thủ và chỉ só tương ứng và thực hiện sắp xếp theo chỉ số
  - Hiển thị kết quả. Lưu ý do sắp xếp tăng dần nên khi lấy top 3 chỉ số cao nhất sẽ phải đảo ngược kết quả

- Kết quả : (ở đây em lấy ví dụ)

3 cầu thủ có age cao nhất

	player	age
486	Ashley Young	38
414	Thiago Silva	38
150	Łukasz Fabiański	38

3 cầu thủ có age thấp nhất

	player	age
87	Leon Chiwome	17
299	Lewis Miley	17
347	David Ozoh	18

2. **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**
   - Ý tưởng:
     - Đầu tiên thực hiện chuyển các thông tin cần tính về dạng số.
     - Sau đó tạo một dictionary với key là tên các đội và ‘all’
     - Sử dụng hàm groupby để tính các thông số cần thiết.
     - Sau đó thêm cái giá trị vào dictionay theo đúng thứ tự
     - Khi thực hiện xong chuyển dictionary thành dataframe với key sẽ là index
   - Code:

```
df2 = df.copy()
index2 = ['team']
df2.fillna(0,inplace=True)
for x in df2.columns[6:]:
    index2.append(x)
    df2[x] = df2[x].astype(float)
df2= df2[index2]
```

```
df2=df2.set_index('team')
```

```
mean = df2.groupby('team').mean()
median =df2.groupby('team').median()
std = df2.groupby('team').std()
mean_all= df2.mean()
median_all= df2.median()
std_all =df2.std()
result2 = dict()
col_result2 = []
result2['all']=[]
for x in mean.index:
    result2[x]=[]
for x in df.columns[6:]:
    col_result2.append(f'Median of {x}')
    col_result2.append(f'Mean of {x}')
    col_result2.append(f'Std of {x}')
    result2['all'].append(median_all[x])
    result2['all'].append(mean_all[x])
    result2['all'].append(std_all[x])
    for y in mean.index:
        result2[y].append(median.loc[y][x])
        result2[y].append(mean.loc[y][x])
        result2[y].append(std.loc[y][x])
result2=pd.DataFrame.from_dict(result2,orient='index',columns=col_result2)
result2
```

- Kết quả thu được:

[6]:

	Median of games	Mean of games	Std of games	Median of games_starts	Mean of games_starts	Std of games_starts	Median of minutes	Mean of minutes	Std of minutes	Median of assists	...	Std of gk_pens_allowed	Median of gk_pens_saved
all	23.0	22.657201	10.136975	16.0	16.941176	11.167179	1419.0	1518.369168	949.241058	1.0	...	0.974582	0.0
Arsenal	27.0	26.809524	10.191266	18.0	19.857143	13.093073	1649.0	1781.571429	1102.745872	2.0	...	0.436436	0.0
Aston Villa	27.0	24.173913	11.109587	20.0	18.130435	12.392462	1652.0	1629.130435	1057.004055	1.0	...	0.208514	0.0
Bournemouth	25.5	22.076923	11.852166	13.0	16.038462	12.732575	1317.5	1438.423077	1074.136832	1.0	...	0.992278	0.0
Brentford	26.0	22.960000	10.346014	15.0	16.720000	10.883933	1321.0	1496.840000	910.122459	1.0	...	0.400000	0.0
Brighton	20.0	20.928571	8.751417	15.0	14.892857	8.603786	1344.5	1338.107143	768.792913	1.0	...	0.786796	0.0
Burnley	16.0	20.392857	9.346575	14.0	14.928571	10.014540	1213.0	1334.857143	851.000706	1.0	...	0.956736	0.0
Chelsea	23.0	21.880000	9.404432	18.0	16.720000	11.066165	1576.0	1495.120000	951.169820	1.0	...	0.439697	0.0
Crystal Palace	22.5	22.458333	9.477567	17.5	17.416667	10.993740	1587.5	1566.000000	910.738831	1.0	...	0.448427	0.0
Everton	28.0	23.304348	11.561829	23.0	18.173913	13.720099	1884.0	1633.173913	1156.480384	0.0	...	1.668115	0.0
Fulham	29.0	27.238095	7.993152	18.0	19.904762	10.084170	1593.0	1773.714286	834.245056	1.0	...	1.745743	0.0
Liverpool	28.0	25.863636	8.993624	17.0	18.954545	8.283531	1671.0	1694.409091	706.077876	2.0	...	0.213201	0.0
Luton Town	23.0	22.840000	9.163696	16.0	16.720000	10.159232	1304.0	1500.840000	865.696188	0.0	...	0.800000	0.0

3. Vẽ histogram phân bố của mỗi chỉ số của tất cả các cầu thủ trong toàn giải và mỗi đội

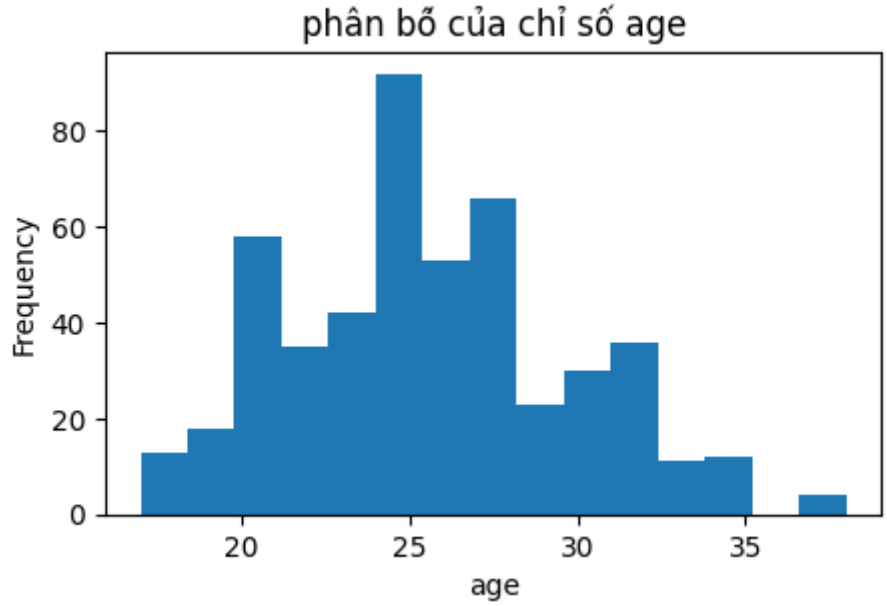
- Ý tưởng:
  - Lấy dữ liệu ra và loại bỏ giá trị NaN. Đối với những chỉ só của thủ môn thì chỉ lấy dữ liệu của thủ môn
  - Thực hiện vẽ histogram với tất cả các trường của bảng
- Code:

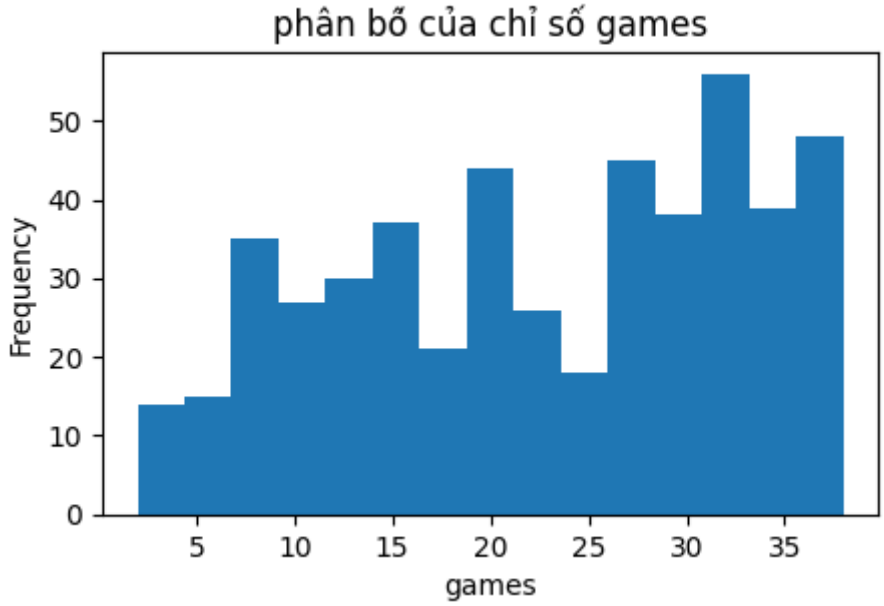
```
df3= df.copy()
print("Phân bố của mỗi chỉ số của các cầu thủ trong toàn giải")
print("")
for i,x in enumerate(df3.columns[5:]):
    df_copy =df.copy()
    if 'gk' in x:
        df_copy= df_copy[df_copy['position']=='GK']
    tmp =df_copy[x]
    tmp.dropna(axis=0,inplace=True)
    plt.figure(figsize=(5,3))
    plt.hist(tmp,bins=15)
    plt.title(f"phân bố của chỉ số {x}")
    plt.xlabel(x)
    plt.ylabel("Frequency")
    plt.show()
```

Phân bố của mỗi chỉ số của các cầu thủ trong toàn giải

```
print("Phân bố của mỗi chỉ số của các cầu thủ trong mỗi đội")
print("")
for y in df['team'].unique():
    df_copy = df.copy()
    df_copy = df_copy[df_copy['team']==y]
    for i,x in enumerate(df_copy.columns[5:]):
        if 'gk' in x:
            df_copy= df_copy[df_copy['position']=='GK']
        tmp = df_copy[x]
        tmp.dropna(axis=0,inplace=True)
        plt.figure(figsize=(5,3))
        plt.hist(tmp,bins=15)
        plt.title(f"Phân bố chỉ số {x} của cầu thủ trong đội bóng {y}")
        plt.xlabel(x)
        plt.ylabel("Frequency")
        plt.show()
```

- Kết quả: (kết quả chi tiết xem tại fille bt2.ipynb)





Bài 3:

I. Phân tích yêu cầu bài toán

- Xác định số nhóm phân loại cầu thủ: Sử dụng thuật toán K-means và phương pháp Elbow hoặc Silhouette Score để chọn số lượng nhóm phù hợp.
- Giảm chiều dữ liệu bằng PCA và vẽ biểu đồ phân cụm.
- Tạo biểu đồ radar để so sánh hai cầu thủ theo các chỉ số đã chọn.

II. Giải quyết vấn đề

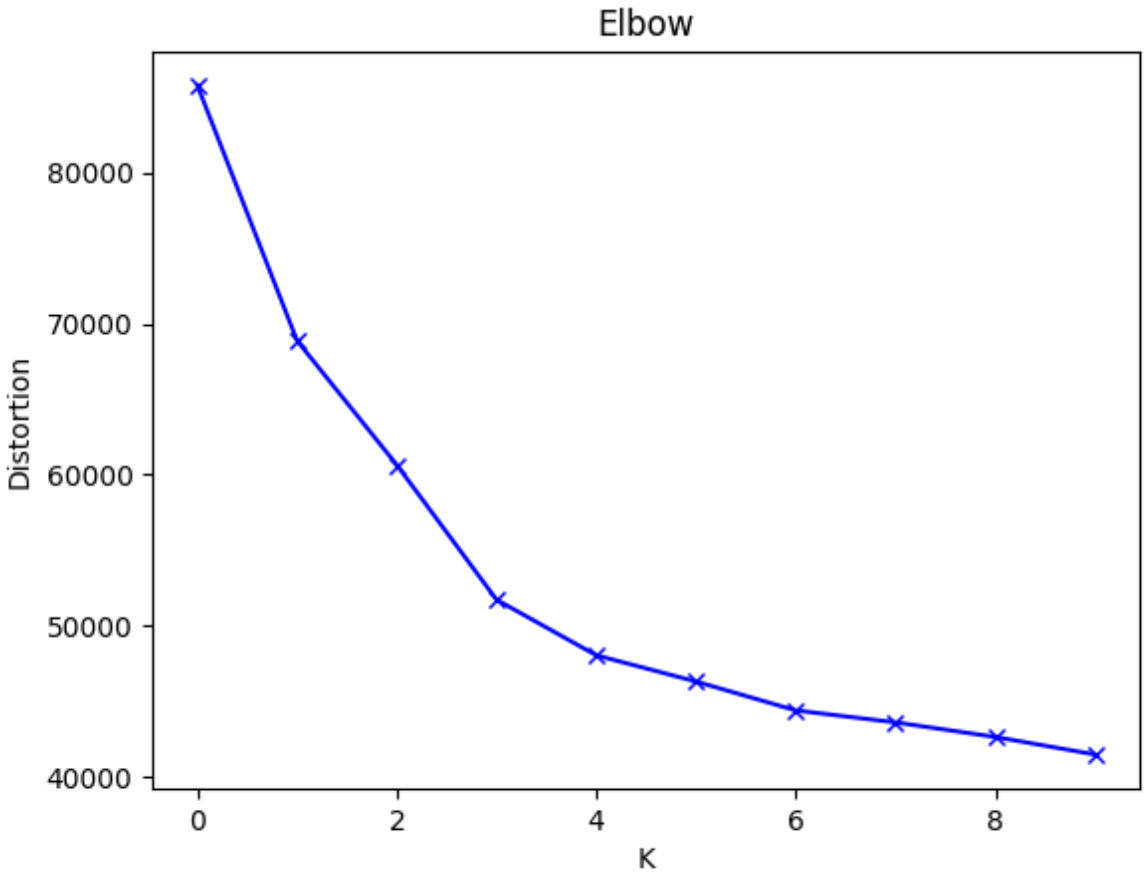
1. Sử dụng Kmean để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau

- Code chuẩn hóa dữ liệu và thực hiện thuật toán elbow

```
df1 = df.copy()
for x in df1.columns:
    if type(x) == str:
        lableEncoder = LabelEncoder()
        df1[x] =lableEncoder.fit_transform(df1[x])
    else:
        df1[x].fillna(df1[x].mean(),inplace=True)
standardScaler = StandardScaler()
df1=standardScaler.fit_transform(df1)
```

```
distortions=[]
for K in range(1,11):
    kmeanModel =KMeans(n_clusters=K)
    kmeanModel.fit(df1)
    distortions.append(kmeanModel.inertia_)
plt.plot(range(10),distortions,'bx-')
plt.xlabel('K')
plt.ylabel('Distortion')
plt.title('Elbow')
plt.show()
```

- Kết quả:



⇨ Từ bảng kết quả trên, Chia các cầu thủ thành 5 nhóm chính.

- Nhận xét: Việc chia cầu thủ thành 5 nhóm khá đúng với thực tế. Trong thế giới bóng đá hiện tại, sẽ có 5 nhóm cầu thủ chính: thủ môn, hậu vệ, tiền vệ phòng ngự, tiền vệ tấn công, tiền đạo.

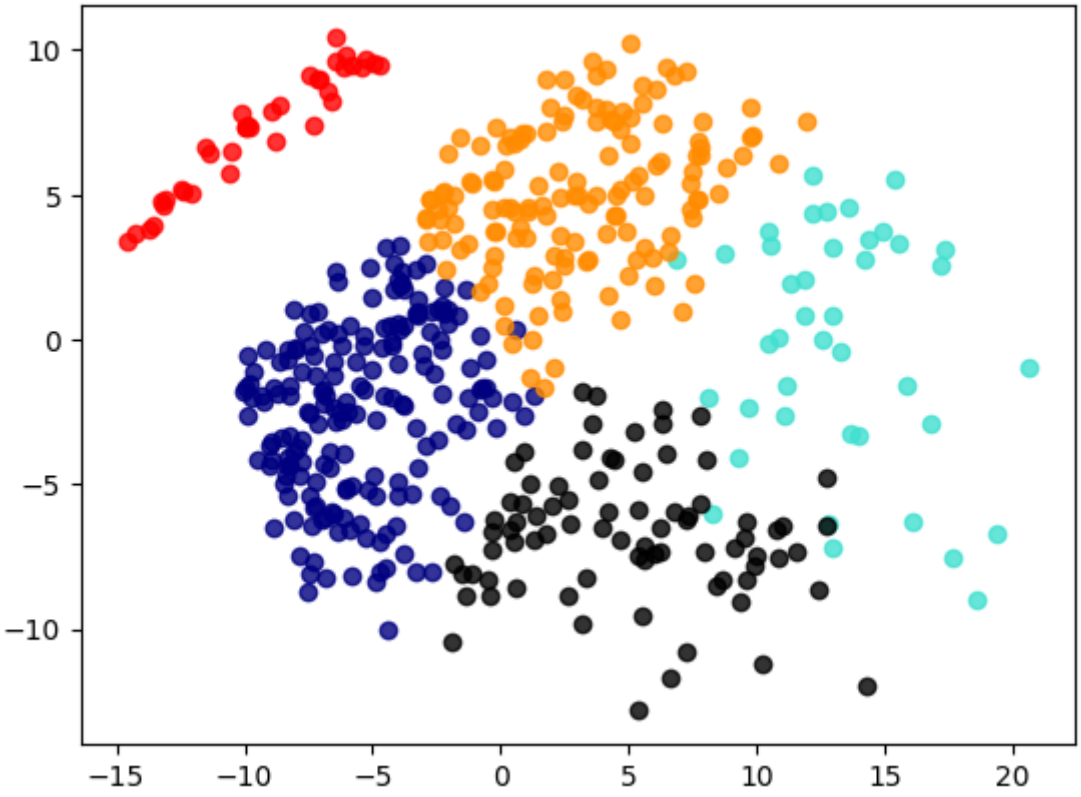
2. Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều và vẽ hình phân cụm các dữ liệu trên mặt 2D

- Code:

```
# phân loại thành 5 cụm
model = KMeans( n_clusters= 5)
ans =model.fit_predict(df1)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_r = pca.fit(df1).transform(df1)

plt.figure()
colors = ['navy','turquoise','darkorange','red','black']

for color, i in zip(colors, [0,1,2,3,4]):
    plt.scatter(X_r[ans == i, 0], X_r[ans == i, 1], color = color, alpha = 0.8, lw = 1)
```

- Kết quả:



3. Viêt chương trình vẽ biểu đồ rada so sánh cầu thủ
   - Code

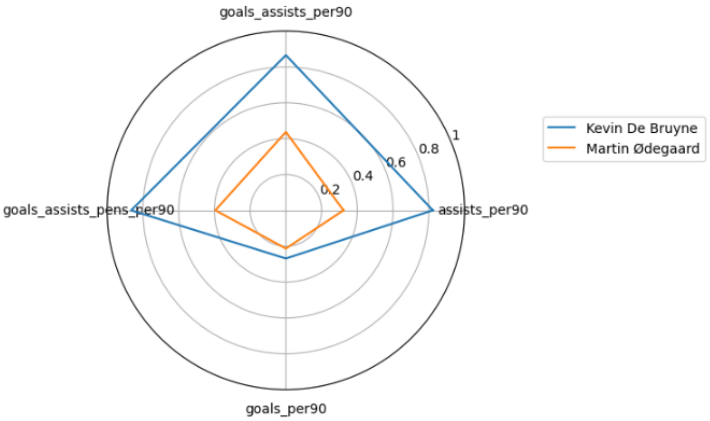
```
import string

# chuẩn hóa min max
def normalize(value:list, max_val:list,min_val:list) -> list:
    target= []
    for min_v,v,max_v in zip(min_val,value,max_val):
        target.append((v-min_v)/(max_v-min_v))
    return target
def draw_radar_chart(player1:string, player2:string, att:list) -> None:

    # lấy giá trị min max các cột
    min_att=[]
    max_att=[]
    for x in att:
        min_att.append(df[x].min())
        max_att.append(df[x].max())
    min_att.append(min_att[0])
    max_att.append(max_att[0])
    # lấy chỉ số 2 cầu thủ và chuẩn hóa chỉ số
    att_p1= np.array(df[df['player']== player1][att])[0]
    att_p1= np.append(att_p1,att_p1[0])
    att1= normalize(att_p1.tolist(),max_att,min_att)

    att_p2= np.array(df[df['player']== player2][att])[0]
    att_p2= np.append(att_p2,att_p2[0])
    att2= normalize(att_p2.tolist(),max_att,min_att)

    # vẽ biểu đồ
    circle = np.linspace(0,2*np.pi,len(att),endpoint=False)
    circle_close= np.append(circle,0)
    ax = plt.subplot(polar=True)
    plt.yticks([0.2,0.4,0.6,0.8,1],['0.2','0.4','0.6','0.8','1'])
    plt.ylim(0,1)
    plt.xticks(circle,att)
    ax.plot(circle_close,att1,label = player1)
    ax.plot(circle_close,att2,label = player2)
    ax.legend(loc='center left',bbox_to_anchor=(1.2,0.7))
    plt.show()
```

   - Ý tưởng:
     - Tìm giá trị min và max của các chỉ số cần so sánh để thực hiện chuẩn hóa min-max
     - Việc chuẩn hóa sẽ giúp hình khi được vẽ ra được tương sứng không bị lệch khi so sánh.
     - Thực hiện vẽ biểu đồ. Circle_close được sử dụng để tạo hình khép kín khi vẽ chỉ số của các cầu thủ

   - Kết quả:

```
draw_radar_chart('Kevin De Bruyne','Martin Ødegaard',['assists_per90', 'goals_assists_per90', 'goals_assists_pens_per90','goals_per90'])
```

```
[np.float64(0.0), np.float64(0.0), np.float64(0.0), np.float64(0.0)] [np.float64(0.9), np.float64(1.19), np.float64(1.19), np.float64(1.
08), np.float64(0.9)]
[0.74 1.03 1.03 0.29 0.74] [0.29 0.52 0.47 0.23 0.29]
```



Bài 4: