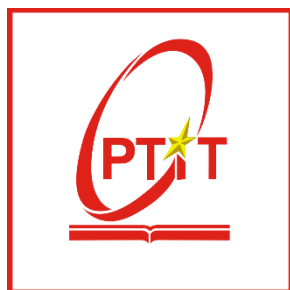


**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LẬP TRÌNH**  
**Môn Học: Python**

**Giảng viên: Kim Ngọc Bách**

**Sinh viên: Nguyễn Hải Nam**

**Mã sinh viên: B22DCCN559**

**Hà Nội – 2024**

*Bài 1. 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/>)*

Để thu thập dữ liệu cầu thủ từ trang fbref.com cho mùa giải 2023-2024 của các đội bóng khác nhau trong Giải Ngoại hạng Anh, ta sử dụng BeautifulSoup (giúp phân tích và lấy dữ liệu html từ web):

```
1 import requests
2 from bs4 import BeautifulSoup as bs
3 import pandas as pd
```

- + requests: Để gửi yêu cầu HTTP và lấy dữ liệu từ các trang web.
- + BeautifulSoup từ bs4: Để phân tích và trích xuất dữ liệu HTML từ trang web.
- + pandas: Để xử lý dữ liệu và chuyển đổi thành DataFrame, thuận tiện cho việc phân tích và lưu trữ vào file CSV.

```
1 for _ in range(0, len(url)):
2     # Gửi yêu cầu và lấy nội dung trang
3     r = requests.get(url[_])
4     r.encoding = 'utf-8'
5     soup = bs(r.content, 'html.parser')
6
7     # Tìm đội và mùa giải
8     main_squad = soup.find('div', {'id': 'info'})
9     squad_tag = main_squad.find('span') if main_squad else None
10
11     # Tìm bảng thống kê chính của đội
12     main_table = soup.find('table', {'id': 'stats_standard_9'})
13     rows = main_table.find_all('tr') if main_table else []
14
15     for row in rows:
16         # Kiểm tra hàng có chứa tên cầu thủ và quốc tịch không
17         name_tag = row.find('th', attrs={'data-stat': 'player'})
18         position_tag = row.find('td', attrs={'data-stat': 'position'})
19         nation_tag = row.find('span', style="white-space: nowrap")
20
```

Sử dụng vòng lặp, lặp qua lần lượt các url dẫn đến các trang thông tin của các đội bóng và lấy về dữ liệu

```
1 if minutes.isdigit() and int(minutes) > 90:
```

Lấy về những cầu thủ có tổng số phút thi đấu trên 90 trong cả mùa giải.

```
1 secondary_table = soup.find('table', {'id': 'stats_keeper_9'})
2 rows_2 = secondary_table.find_all('tr') if secondary_table else []
3 found = False
4 for row_2 in rows_2:
5     # Kiểm tra hàng có chứa tên cầu thủ không
6     name_tag_2 = row_2.find('th', attrs={'data-stat': 'player'})
7     if name_tag_2 and name_tag_2.text.strip() == name:
8         # Lấy tất cả các cột từ cột 8 đến cột cuối cùng
9         columns_2 = row_2.find_all('td')[7:-1] # Cột từ thứ 8 đến cuối
10        additional_data2 = [col.text.strip() for col in columns_2]
11        additional_data2 = [(col.text.strip() if col.text.strip() else 'N/A') for col in columns_2]
12
13
14        # Kết hợp thông tin cầu thủ với dữ liệu từ bảng phụ
15        player_info += additional_data2
16        found = True
17        break
18    if not found:
19        player_info += ['N/A'] * 15
```

Lấy thêm dữ liệu ở các bảng phụ có trong web, mỗi bảng được lấy các cột cần thiết và thêm vào danh sách player\_info của cầu thủ.

```
1 df = pd.DataFrame(player_list, columns=['Name', 'Squad', 'Nation', 'Position', 'Age', 'MP', 'Starts', 'Min', '90s', 'Gls', 'Ast', 'G+A',
2     'G-PK', 'PK', 'PKatt', 'CrdY', 'Crdr', 'xG', 'npxG', 'xAG', 'npxG+xAG', 'PrnC',
3     'PrnP', 'PrGR', 'Gls', 'Ast', 'G+A', 'G-PK', 'G+A-PK', 'xG', 'xAG', 'xG+xAG', 'npxG', 'npxG+xAG', 'GA',
4     'GA90', 'SoTA', 'Saves', 'Save%', 'W', 'D', 'L', 'CS', 'CS%', 'PKatt', 'PKA', 'PKsv', 'PKm', 'Save%'])
```

```
1 df.to_csv('results.csv', index=False)
```

Sau khi thu thập tất cả dữ liệu, danh sách player\_list được chuyển thành DataFrame (df) với các cột tương ứng. Sau đó được lưu vào tệp CSV (results.csv).

file: results.csv.

	Name	Squad	Nation	Position	Age	MP	Starts	Min	90s	Gls	Ast	G+A	G-PK	PK	PKatt	CrdY	CrdR	xG	npG	xAG	npG+xAG
2	Aaron Cresswell	WestHamUnited	ENG	DF,FW	33	11	4	436	4.8	0	0	0	0	0	0	1	0	0.0	0.0	0.4	0.4
3	Aaron Hickey	Brentford	SCO	DF	21	9	9	713	7.9	0	0	0	0	0	0	5	0	0.2	0.2	0.1	0.3
4	Aaron Ramsdale	Arsenal	ENG	GK	25	6	6	540	6.0	0	0	0	0	0	0	0	0	0.0	0.0	0.0	0.0
5	Aaron Ramsey	Burnley	ENG	MF,FW	20	14	5	527	5.9	0	0	0	0	0	0	1	0	0.3	0.3	0.4	0.7
6	Aaron Wan-Bissaka	ManchesterUnited	ENG	DF	25	22	20	1,780	19.8	0	2	2	0	0	0	4	0	0.1	0.1	1.5	1.6
7	Abdoulaye Doucouré	Everton	MLI	FW,MF	30	32	32	2,629	29.2	7	1	8	7	0	0	7	0	8.8	8.8	2.9	11.6
8	Adam Lallana	Brighton&HoveAlbion	ENG	MF,FW	35	25	13	850	9.4	0	1	1	0	0	0	2	0	0.8	0.8	1.7	2.5
9	Adam Smith	Bournemouth	ENG	DF	32	28	25	2,150	23.9	0	2	2	0	0	0	6	0	0.1	0.1	1.3	1.4
10	Adam Webster	Brighton&HoveAlbion	ENG	DF	28	15	13	1,144	12.7	0	0	0	0	0	0	2	0	0.4	0.4	0.1	0.6
11	Adam Wharton	CrystalPalace	ENG	MF	19	16	15	1,297	14.4	0	3	3	0	0	0	2	0	0.3	0.3	2.4	2.8
12	Adama Traoré	Fulham	ESP	FW,MF	27	17	1	377	4.2	2	3	5	2	0	0	2	0	1.5	1.5	0.7	2.2
13	Albert Sambi Lokonga	LutonTown	BEL	MF	23	17	16	1,303	14.5	1	3	4	1	0	0	4	0	0.6	0.6	1.4	2.0
14	Alejandro Garnacho	ManchesterUnited	ARG	FW	19	36	30	2,565	28.5	7	4	11	7	0	0	4	0	8.4	8.3	5.1	13.4
15	Alex Iwobi	Everton	NGA	MF	27	2	2	140	1.6	0	0	0	0	0	0	0	0	0.3	0.3	0.2	0.5

(...)

Bài 2:

*\*tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số:*

```
1 import pandas as pd
2
3 df = pd.read_csv('results.csv')
4 df['Min'] = df['Min'].str.replace(',', '').astype(int)
5 df.to_csv('results.csv', index=False)
6 df = pd.read_csv('results.csv')
7 for column in df.columns[4:]:
8     try:
9         df[column] = pd.to_numeric(df[column], errors='coerce')
10
11         # Sắp xếp theo cột hiện tại theo thứ tự giảm dần và lấy 3 hàng đầu
12         top_3 = df.sort_values(by=column, ascending=False).head(3)
13
14         # Sắp xếp theo cột hiện tại theo thứ tự tăng dần và lấy 3 hàng cuối (loại trừ NaN)
15         bottom_3 = df.sort_values(by=column, ascending=True).dropna(subset=[column]).head(3)
16
17         print(f"Top 3 cho {column}: \n{top_3[['Name', column]]}\n")
18         print(f"3 cầu thủ cuối bảng cho {column}: \n{bottom_3[['Name', column]]}\n")
19     except (TypeError, ValueError): # Xử lý các cột không thể chuyển đổi thành số
20         print(f"Bỏ qua cột '{column}' (kiểu dữ liệu không phải số hoặc hỗn hợp)\n")
```

## 1, Đọc và xử lý dữ liệu:

Sử dụng thư viện pandas để đọc dữ liệu từ tệp results.csv.

Cột Min (phút chơi) được chuyển đổi từ kiểu chuỗi thành số nguyên bằng cách loại bỏ dấu phẩy (,) nếu có, và sau đó được ghi đè vào results.csv dưới dạng số.

## 2, Chuyển đổi dữ liệu sang kiểu số:

Đối với mỗi cột từ cột thứ 4 trở đi (bỏ qua các cột thông tin cầu thủ như Name, Squad, Nation, Position), chương trình cố gắng chuyển đổi dữ liệu trong cột đó thành kiểu số (numeric) để có thể thực hiện phép so sánh.

Nếu có bất kỳ giá trị nào không thể chuyển đổi (ví dụ: dữ liệu không phải là số hoặc hỗn hợp), pd.to\_numeric() sẽ thay thế bằng giá trị NaN (errors='coerce').

## 3, Tìm cầu thủ top 3 và bottom 3 cho từng cột:

Top 3: Với mỗi cột đã được chuyển đổi thành kiểu số, chương trình sắp xếp dữ liệu theo thứ tự giảm dần và lấy 3 cầu thủ đứng đầu cho cột đó.

Bottom 3: Chương trình cũng sắp xếp dữ liệu theo thứ tự tăng dần và lấy 3 cầu thủ xếp cuối, loại trừ các giá trị NaN (bỏ qua các cầu thủ không có dữ liệu cho cột đó).

Kết quả cho mỗi cột sẽ được in ra, bao gồm tên cầu thủ và giá trị của cột hiện tại.

## 4, Xử lý ngoại lệ:

Nếu một cột không thể chuyển đổi sang số (ví dụ như các cột không chứa dữ liệu số), chương trình sẽ bỏ qua và in thông báo rằng cột đó không phải là dữ liệu số hoặc là hỗn hợp.

Kết quả:

Top 3 cầu thủ cao nhất, thấp nhất ở các chỉ số:

Top 3 cho Age:		Top 3 cho Starts:		Top 3 cho Gls:	
	Name Age		Name Starts		Name Gls
492	Łukasz Fabiański 38	210	James Tarkowski 38	149	Erling Haaland 27
446	Thiago Silva 38	62	Bernd Leno 38	101	Cole Palmer 22
47	Ashley Young 38	478	William Saliba 38	16	Alexander Isak 21
3 cầu thủ cuối bảng cho Age:		3 cầu thủ cuối bảng cho Starts:		3 cầu thủ cuối bảng cho Gls:	
	Name Age		Name Starts		Name Gls
277	Leon Chiwome 17	324	Matt Ritchie 0	0	Aaron Cresswell 0
284	Lewis Miley 17	191	Ivan Perišić 0	236	John Fleck 0
482	Wilson Odobert 18	226	Jesurun Rak Sakyi 0	239	Jonny Evans 0
Top 3 cho MP:		Top 3 cho Min:		Top 3 cho Ast:	
	Name MP		Name Min		Name Ast
228	Joachim Andersen 38	33	André Onana 3420	378	Ollie Watkins 13
84	Carlton Morris 38	243	Jordan Pickford 3420	101	Cole Palmer 11
243	Jordan Pickford 38	478	William Saliba 3420	69	Brennan Johnson 10
3 cầu thủ cuối bảng cho MP:		3 cầu thủ cuối bảng cho Min:		3 cầu thủ cuối bảng cho Ast:	
	Name MP		Name Min		Name Ast
320	Matheus Nunes 2	236	John Fleck 92	0	Aaron Cresswell 0
359	Neal Maupay 2	259	Kalvin Phillips 93	281	Lewis Dobbin 0
188	Ionuț Radu 2	191	Ivan Perišić 103	278	Lesley Ugochukwu 0

(...)

*\* 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*

```

1 import pandas as pd
2
3 df = pd.read_csv('results.csv')
4
5 # Lấy danh sách các cột chỉ số (loại trừ 'Name' và 'Squad')
6 attribute_cols = df.select_dtypes(include=['number']).columns.tolist()
7
8 # Tạo một DataFrame rỗng với hàng là các đội và 'all' cho toàn bộ giải đấu
9 teams = ['all'] + df['Squad'].unique().tolist()
10 results = pd.DataFrame(index=teams)
11
12 for col in attribute_cols:
13     # Tính cho toàn bộ giải đấu
14     results.at['all', f'{col}_Median'] = df[col].median()
15     results.at['all', f'{col}_Mean'] = df[col].mean()
16     results.at['all', f'{col}_Std'] = df[col].std()
17
18     # Tính cho từng đội
19     for squad in df['Squad'].unique():
20         results.at[squad, f'{col}_Median'] = df[df['Squad'] == squad][col].median()
21         results.at[squad, f'{col}_Mean'] = df[df['Squad'] == squad][col].mean()
22         results.at[squad, f'{col}_Std'] = df[df['Squad'] == squad][col].std()
23
24 results.to_csv('results2.csv')

```

### 1, Đọc dữ liệu từ tệp CSV:

Sử dụng pandas để đọc dữ liệu từ results.csv vào DataFrame df.

### 2, Lấy danh sách các cột chỉ số:

Tìm tất cả các cột có dữ liệu kiểu số (loại trừ các cột tên cầu thủ và đội) bằng cách dùng `select_dtypes(include=['number'])`. `attribute_cols` sẽ là danh sách các cột chứa các chỉ số cần tính toán (ví dụ: số bàn thắng, số phút chơi, số trận đấu, v.v.).

### 3, Tạo DataFrame results để lưu kết quả:

Khởi tạo results, một DataFrame rỗng với chỉ số (index) là danh sách các đội (được lấy từ cột Squad) và một hàng đặc biệt là 'all', đại diện cho toàn bộ giải đấu.

Các kết quả thống kê sẽ được tính toán và lưu trong results.

### 4, Tính toán trung vị, trung bình và độ lệch chuẩn:

Cho toàn bộ giải đấu ('all'):

Dùng `median()`, `mean()`, và `std()` để tính lần lượt trung vị, trung bình và độ lệch chuẩn cho từng cột chỉ số. Kết quả được lưu vào hàng 'all' của results.

Cho từng đội:

Lặp qua mỗi đội trong `df['Squad'].unique()`.

Tính toán các giá trị trung vị, trung bình và độ lệch chuẩn cho từng chỉ số trong mỗi đội bằng cách lọc các hàng thuộc về đội đó (`df[df['Squad'] == squad][col]`).

Kết quả của mỗi đội được lưu vào các cột tương ứng trong hàng của đội đó trong results.

### 5, Lưu kết quả vào tệp CSV:

Kết quả tính toán được lưu vào tệp results2.csv.

*\* Vẽ histogram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội.*

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 # Đọc dữ liệu từ file CSV
6 df = pd.read_csv('results.csv')
7
8 # Xác định các chỉ số cụ thể để vẽ histogram
9 stat_columns = ['Age', 'MP', 'Starts', '90s', 'Gls', 'Ast']
10
11 # 1. Vẽ histogram phân bố cho mỗi chỉ số trên toàn giải
12 for col in stat_columns:
13     plt.figure(figsize=(10, 6))
14     sns.histplot(df[col].dropna(), kde=True)
15     plt.title(f'Phân bố {col} - Toàn giải')
16     plt.xlabel(col)
17     plt.ylabel('Tần suất')
18     plt.show()
19
20 # 2. Vẽ histogram phân bố cho mỗi chỉ số cho từng đội
21 teams = df['Squad'].unique()
22 for team in teams:
23     team_data = df[df['Squad'] == team]
24     for col in stat_columns:
25         plt.figure(figsize=(10, 6))
26         sns.histplot(team_data[col].dropna(), kde=True)
27         plt.title(f'Phân bố {col} - {team}')
28         plt.xlabel(col)
29         plt.ylabel('Tần suất')
30         plt.show()
```

1, Khởi tạo các thư viện và đọc dữ liệu từ results.csv:

Import các thư viện pandas, matplotlib.pyplot, seaborn.

Đọc dữ liệu từ results.csv vào DataFrame df, chứa thông tin chi tiết về các chỉ số cầu thủ.

2, Xác định các chỉ số cần vẽ biểu đồ:

ở đây, ta chỉ lấy 1 số các chỉ số để thấy của cầu thủ để vẽ histogram.

stat\_columns là danh sách các cột chứa các chỉ số cần phân tích, bao gồm: 'Age' (tuổi), 'MP' (số trận), 'Starts' (số trận đá chính), '90s' (số lần chơi đủ 90 phút), 'Gls' (số bàn thắng), và 'Ast' (số kiến tạo).

3, Vẽ biểu đồ phân bố trên toàn giải đấu:

Vòng lặp for để lần lượt tạo biểu đồ cho mỗi cột chỉ số trong stat\_columns.

4, Thiết lập biểu đồ:

Với plt.figure(figsize=(10, 6)), đặt kích thước biểu đồ là 10x6 inch.



`sns.histplot(df[col].dropna(), kde=True)` tạo biểu đồ histogram cho từng cột chỉ số, loại bỏ các giá trị NaN. `kde=True` thêm đường cong mật độ để dễ quan sát phân bố.

5, Thiết lập nhãn và tiêu đề:

`plt.title(f'Phân bố {col} - Toàn giải')` đặt tiêu đề biểu đồ theo tên chỉ số và chỉ rõ rằng đây là phân bố cho toàn giải.

`plt.xlabel(col)` và `plt.ylabel('Tần suất')` đặt nhãn trục x và y lần lượt là tên chỉ số và "Tần suất".

6, Hiện thị biểu đồ với `plt.show()`.

7, Vẽ biểu đồ phân bố cho từng đội:

+ Lấy danh sách các đội bằng `teams = df['Squad'].unique()` (mỗi đội chỉ xuất hiện một lần trong danh sách).

+ Lặp qua từng đội:

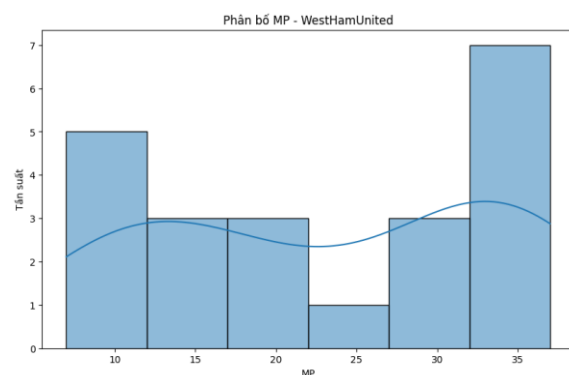
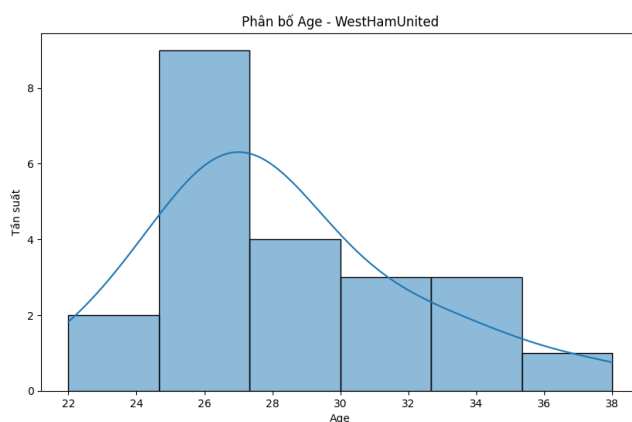
Lọc dữ liệu của từng đội với `team_data = df[df['Squad'] == team]`.

Tạo một vòng lặp bên trong để vẽ histogram cho từng cột chỉ số trong `stat_columns` cho từng đội.

Thiết lập biểu đồ và nhãn tương tự như phần trước, nhưng tiêu đề được điều chỉnh để hiển thị tên đội: `plt.title(f'Phân bố {col} - {team}')`.

+ Hiện thị từng biểu đồ với `plt.show()`.

(một số histogram đã vẽ: )



*\* 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*

```
1 import pandas as pd
2
3 df = pd.read_csv('results.csv')
4 for column in df.columns[5:]: # Bắt đầu từ cột 5 ('MP')
5     try:
6         max_value = df[column].max() # Tìm giá trị lớn nhất
7         top_teams = df[df[column] == max_value] # Lọc các hàng có giá trị lớn nhất
8         team_names = top_teams['Squad'].unique() # Lấy tên đội duy nhất
9         print(f"Đội bóng có chỉ số {column} cao nhất: {team_names}")
10    except TypeError: # Xử lý các cột không phải số
11        print(f"Bỏ qua cột '{column}' (kiểu dữ liệu không phải số)")
12
```

1, Đọc dữ liệu từ file results.csv.

2, Lặp qua các cột để tìm chỉ số cao nhất:

Vòng lặp for duyệt qua các cột bắt đầu từ cột thứ 5 (df.columns[5:]), giả định cột thứ 5 (MP) trở đi là các chỉ số cầu thủ.

3, Tìm đội có chỉ số cao nhất:

max\_value = df[column].max() tìm giá trị lớn nhất của cột hiện tại.

top\_teams = df[df[column] == max\_value] lọc các hàng có giá trị bằng max\_value, nhằm tìm các cầu thủ hoặc đội đạt giá trị cao nhất ở chỉ số đó.

team\_names = top\_teams['Squad'].unique() lấy danh sách tên đội duy nhất từ các hàng có giá trị cao nhất.

4, In kết quả:

print(f"Đội bóng có chỉ số {column} cao nhất: {team\_names}") hiển thị tên đội có giá trị lớn nhất cho mỗi chỉ số.

5, Xử lý lỗi cho các cột không phải số:

Khởi try-except xử lý trường hợp cột không thể chuyển đổi thành số (TypeError), bỏ qua các cột kiểu dữ liệu không phù hợp và in thông báo.

Kết quả: (dưới đây là 1 số các chỉ số với các đội đứng đầu ở các chỉ số đó)

```
Đội bóng có chỉ số MP cao nhất: ['ManchesterUnited' 'Fulham' 'LutonTown' 'Arsenal' 'Bournemouth'
'TottenhamHotspur' 'Everton' 'CrystalPalace' 'WolverhamptonWanderers'
'AstonVilla' 'Brentford']
Đội bóng có chỉ số Starts cao nhất: ['ManchesterUnited' 'Fulham' 'TottenhamHotspur' 'Everton' 'CrystalPalace'
'WolverhamptonWanderers' 'LutonTown' 'Arsenal']
Đội bóng có chỉ số Min cao nhất: ['ManchesterUnited' 'Fulham' 'TottenhamHotspur' 'Everton'
'WolverhamptonWanderers' 'Arsenal']
Đội bóng có chỉ số 90s cao nhất: ['ManchesterUnited' 'Fulham' 'TottenhamHotspur' 'Everton'
'WolverhamptonWanderers' 'Arsenal']
Đội bóng có chỉ số Gls cao nhất: ['ManchesterCity']
Đội bóng có chỉ số Ast cao nhất: ['AstonVilla']
Đội bóng có chỉ số G+A cao nhất: ['Chelsea']
Đội bóng có chỉ số G-PK cao nhất: ['ManchesterCity']
Đội bóng có chỉ số PK cao nhất: ['Chelsea']
Đội bóng có chỉ số PKatt cao nhất: ['Chelsea']
Đội bóng có chỉ số CrdY cao nhất: ['Fulham' 'Bournemouth']
Đội bóng có chỉ số CrdR cao nhất: ['SheffieldUnited' 'Chelsea' 'TottenhamHotspur']
Đội bóng có chỉ số xG cao nhất: ['ManchesterCity']
Đội bóng có chỉ số npvG cao nhất: ['ManchesterCity']
Đội bóng có chỉ số xAG cao nhất: ['ManchesterUnited' 'Liverpool' 'TottenhamHotspur']
Đội bóng có chỉ số npvG+xAG cao nhất: ['Liverpool']
Đội bóng có chỉ số PrgC cao nhất: ['ManchesterCity']
Đội bóng có chỉ số PrgP cao nhất: ['ManchesterCity']
Đội bóng có chỉ số PrgR cao nhất: ['Arsenal']
```

+ Theo em, ở mùa giải 2023-2024 premier league, thì **Tottenham Hotspur** có phong độ tốt nhất, các chỉ số cầu thủ cho thấy Tottenham luôn đứng ở top cao, đội bóng có phong độ ổn định, luôn duy trì vị trí cao trên bảng xếp hạng.

### Bài 3:

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

```
1 import pandas as pd
2 from sklearn.cluster import KMeans
3 from sklearn.preprocessing import StandardScaler
4
5 df = pd.read_csv('results.csv')
6
7 features = ['Age', 'MP', 'Starts', 'Min', '90s', 'Gls', 'Ast'] # Thay đổi các features nếu cần
8 X = df[features]
9
10 # Convert 'Min' column to numeric, handling errors by setting non-numeric values to NaN
11 X['Min'] = pd.to_numeric(X['Min'].str.replace(',', ''), errors='coerce')
12
13 scaler = StandardScaler()
14 X_scaled = scaler.fit_transform(X)
15 n_clusters = 5 # Thay đổi số lượng clusters nếu cần
16
17 kmeans = KMeans(n_clusters=n_clusters, random_state=42) # random_state để đảm bảo kết quả nhất quán
18 kmeans.fit(X_scaled)
19
20 df['Cluster'] = kmeans.labels_
21
22 print(df[df['Cluster'] == 0])
23
24 # Calculate cluster means, ensuring only numeric columns are used
25 cluster_means = df.groupby('Cluster')[features].mean(numeric_only=True)
26 print(cluster_means)
27
28
29
30 df.to_csv('results3.csv', index=False)
```

#### 1, Đọc và chuẩn hóa dữ liệu:

Đọc dữ liệu từ results.csv.

Xác định các thuộc tính (features) sử dụng để phân cụm: Age, MP, Starts, Min, 90s, Gls, Ast. (ở đây em chọn ra 7 thuộc tính này để phân cụm)

Chuyển cột Min về dạng số, loại bỏ các dấu phẩy, đồng thời thay các giá trị không hợp lệ bằng NaN.

#### 2, Chuẩn hóa dữ liệu với StandardScaler:

Sử dụng StandardScaler để chuẩn hóa các chỉ số, nhằm đưa chúng về cùng một thang đo và tránh tình trạng các thuộc tính có giá trị lớn (như Min) làm ảnh hưởng đến quá trình phân cụm.

### 3, Áp dụng thuật toán KMeans:

Áp dụng thuật toán phân cụm KMeans với số cụm (n\_clusters) là 5, có thể thay đổi nếu bạn muốn thử nghiệm với số cụm khác. random\_state=42 giúp đảm bảo kết quả nhất quán qua các lần chạy.

Kết quả phân cụm (nhãn cụm) sẽ được lưu vào cột Cluster trong DataFrame.

### 4, Hiển thị và tính toán trung bình cho từng cụm:

In các cầu thủ trong cụm đầu tiên (Cluster = 0) để xem xét kết quả phân cụm.

Tính trung bình cho từng thuộc tính trong từng cụm và lưu vào biến cluster\_means. Chỉ các cột dạng số (numeric\_only=True) sẽ được tính trung bình.

### 5, Lưu kết quả:

Kết quả cuối cùng, bao gồm cả nhãn cụm cho mỗi cầu thủ, sẽ được lưu vào results3.csv.

	Age	MP	Starts	90s	Gls	Ast
Cluster						
0	24.700000	26.020000	18.446667	18.235333	2.926667	1.893333
1	22.256198	11.537190	4.958678	5.503306	0.479339	0.396694
2	24.960784	34.333333	30.745098	29.801961	9.862745	7.117647
3	30.045977	15.091954	8.643678	9.002299	0.678161	0.551724
4	27.214286	33.416667	31.726190	31.102381	1.619048	1.345238

on	OG	Recov	Won	Lost.1	Won%	Cluster
	0	18	6	3	66.7	3
	0	42	1	9	10.0	1
	0	6	0	0		1
	0	24	4	5	44.4	1
	0	94	21	19	52.5	0
	0	144	24	40	37.5	4
	0	22	3	5	37.5	3
	0	92	18	22	45.0	4
	1	63	33	13	71.7	3
	0	76	7	12	36.8	0

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

```
1 import pandas as pd
2 from sklearn.cluster import KMeans
3 from sklearn.preprocessing import StandardScaler
4 from sklearn.decomposition import PCA
5 import matplotlib.pyplot as plt
6
7 # Đọc dữ liệu
8 df = pd.read_csv('results.csv')
9
10 # Xác định các thuộc tính sử dụng để phân cụm
11 features = ['Age', 'MP', 'Starts', 'Min', '90s', 'Gls', 'Ast', 'G+A', 'G-PK', 'PK', 'PKatt', 'CrdY', 'CrdR', 'xG', 'npxG', 'xAG', 'npxG+xAG', 'PrG',
12            'PrGR', 'PrGR', 'Gls', 'Ast', 'G+A', 'G-PK', 'G+A-PK', 'xG', 'xAG', 'xG+xAG', 'npxG', 'npxG+xAG']
13 X = df[features]
14
15 # Chuyển cột 'Min' thành kiểu số, loại bỏ dấu phẩy và xử lý lỗi
16 X['Min'] = pd.to_numeric(X['Min'].str.replace(',', ''), errors='coerce')
17
18 # Kiểm tra và điền giá trị thiếu nếu có
19 X = X.fillna(0) # Điền giá trị NaN bằng 0, hoặc bạn có thể dùng các phương pháp xử lý khác
20
21 # Chuẩn hóa dữ liệu
22 scaler = StandardScaler()
23 X_scaled = scaler.fit_transform(X)
24
25 # Áp dụng KMeans clustering
26 n_clusters = 5
27 kmeans = KMeans(n_clusters=n_clusters, random_state=42)
28 kmeans.fit(X_scaled)
29 df['Cluster'] = kmeans.labels_
30
31 # Áp dụng PCA để giảm xuống 2 chiều
32 pca = PCA(n_components=2)
33 X_pca = pca.fit_transform(X_scaled)
34
35 # Vẽ biểu đồ phân tán các cụm
36 plt.figure(figsize=(8, 6))
37 plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df['Cluster'], cmap='viridis')
38 plt.title('Clusters of Football Players (PCA)')
39 plt.xlabel('Principal Component 1')
40 plt.ylabel('Principal Component 2')
41 plt.show()
```

## 1, Đọc và chọn dữ liệu:

Đọc dữ liệu từ file results.csv.

Chọn các thuộc tính (biến) cần dùng cho phân cụm trong features.

## 2, Xử lý và chuẩn hóa dữ liệu:

Chuyển cột Min sang kiểu số, loại bỏ dấu phẩy và thay các giá trị không hợp lệ (N/A) bằng 0.

Chuẩn hóa dữ liệu bằng StandardScaler để đưa các thuộc tính về cùng thang đo, giúp thuật toán KMeans hoạt động hiệu quả hơn.

## 3, Phân cụm với KMeans:

Áp dụng KMeans với 5 cụm (n\_clusters=5) và gán nhãn cụm vào cột mới Cluster của DataFrame.

Mỗi cầu thủ sẽ được gán vào một cụm dựa trên đặc điểm của họ.

#### 4, Giảm chiều dữ liệu bằng PCA:

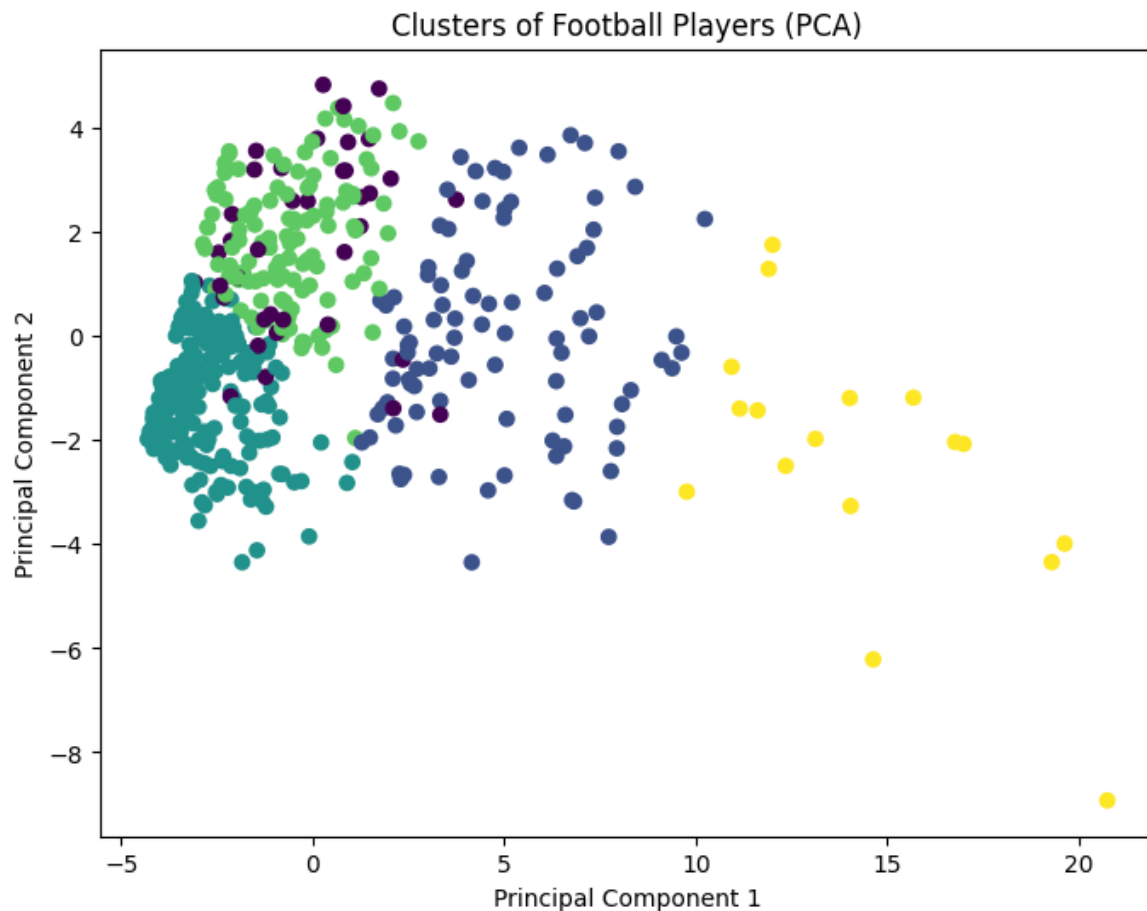
Dùng PCA để giảm số chiều của dữ liệu từ nhiều thuộc tính xuống 2 thành phần chính ( $n\_components=2$ ). Điều này giúp bạn có thể trực quan hóa các cụm trong không gian 2D.

#### 5, Vẽ biểu đồ phân tán:

Vẽ biểu đồ phân tán 2D, trong đó các cầu thủ được hiển thị dưới dạng các điểm màu sắc khác nhau dựa trên cụm của họ.

Biểu đồ này cho thấy cách các cầu thủ được phân chia thành các cụm khác nhau dựa trên đặc điểm thống kê của họ.

Kết quả:



*\*Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ*

```
1 import argparse
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from math import pi
6
7 # Hàm để vẽ biểu đồ radar
8 def radar_chart(df, player1, player2, attributes):
9     # Lấy dữ liệu của hai cầu thủ
10    player1_data = df[df['Player'] == player1].iloc[0][attributes].values
11    player2_data = df[df['Player'] == player2].iloc[0][attributes].values
12
13    # Tạo các góc cho biểu đồ radar
14    num_vars = len(attributes)
15    angles = [n / float(num_vars) * 2 * pi for n in range(num_vars)]
16    angles += angles[:1] # Đóng vòng tròn
17
18    # Chuyển dữ liệu cầu thủ thành vòng tròn
19    player1_data = np.append(player1_data, player1_data[0])
20    player2_data = np.append(player2_data, player2_data[0])
21
22    # Vẽ biểu đồ radar
23    fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
24    plt.xticks(angles[:-1], attributes, color='grey', size=8)
25
26    # Vẽ dữ liệu của cầu thủ thứ nhất
27    ax.plot(angles, player1_data, linewidth=1, linestyle='solid', label=player1)
28    ax.fill(angles, player1_data, 'b', alpha=0.1)
29
30    # Vẽ dữ liệu của cầu thủ thứ hai
31    ax.plot(angles, player2_data, linewidth=1, linestyle='solid', label=player2)
32    ax.fill(angles, player2_data, 'r', alpha=0.1)
33
34    # Cài đặt tên cầu thủ trong phần chú thích
35    plt.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))
36    plt.title(f"So sánh cầu thủ {player1} và {player2}")
37    plt.show()
38
39 # Hàm chính để xử lý tham số đầu vào
40 def main():
41     parser = argparse.ArgumentParser(description="Vẽ biểu đồ radar để so sánh các chỉ số của hai cầu thủ.")
42     parser.add_argument("--p1", required=True, help="Tên cầu thủ thứ nhất")
43     parser.add_argument("--p2", required=True, help="Tên cầu thủ thứ hai")
44     parser.add_argument("--Attribute", required=True, help="Danh sách các chỉ số cần so sánh, cách nhau bởi dấu phẩy")
45
46     args = parser.parse_args()
47     player1 = args.p1
48     player2 = args.p2
49     attributes = args.Attribute.split(',')
50
51     # Đọc dữ liệu từ file CSV
52     df = pd.read_csv('results.csv')
53
54     # Kiểm tra các cầu thủ có tồn tại trong dữ liệu hay không
55     if player1 not in df['Player'].values or player2 not in df['Player'].values:
56         print(f"Không tìm thấy dữ liệu của {player1} hoặc {player2}.")
57         return
58
59     # Kiểm tra các thuộc tính có tồn tại trong dữ liệu hay không
60     missing_attributes = [attr for attr in attributes if attr not in df.columns]
61     if missing_attributes:
62         print(f"Các thuộc tính không tồn tại trong dữ liệu: {', '.join(missing_attributes)}")
63         return
64
65     # Vẽ biểu đồ radar
66     radar_chart(df, player1, player2, attributes)
67
68 if __name__ == "__main__":
69     main()
```



Về chi tiết đoạn code:

1, Hàm `radar_chart(df, player1, player2, attributes)`:

Lấy dữ liệu của hai cầu thủ được chọn theo tên (`player1` và `player2`) và các thuộc tính cần so sánh (`attributes`).

Tạo các góc cho biểu đồ radar dựa trên số lượng thuộc tính, sau đó đóng vòng tròn bằng cách thêm lại góc đầu tiên vào cuối danh sách.

Vẽ đường biểu diễn và tô màu cho khu vực radar của mỗi cầu thủ. Cầu thủ thứ nhất có màu xanh nhạt và cầu thủ thứ hai có màu đỏ nhạt.

Thêm chú thích và tiêu đề để phân biệt cầu thủ và biểu diễn.

2, Hàm `main()`:

Sử dụng `argparse` để lấy đầu vào từ dòng lệnh, bao gồm:

--p1 và --p2: tên của hai cầu thủ.

--Attribute: danh sách các thuộc tính cần so sánh, cách nhau bởi dấu phẩy.

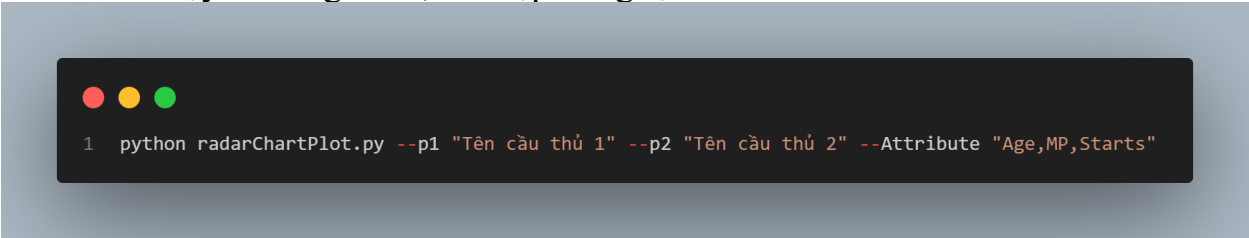
Đọc dữ liệu từ file CSV (`results.csv`).

Kiểm tra xem hai cầu thủ và các thuộc tính có tồn tại trong dữ liệu không. Nếu không, thông báo lỗi sẽ hiển thị.

Gọi hàm `radar_chart()` để vẽ biểu đồ radar.

3, Chạy chương trình:

Để chạy chương trình, ta nhập dòng lệnh sau:



```
1 python radarChartPlot.py --p1 "Tên cầu thủ 1" --p2 "Tên cầu thủ 2" --Attribute "Age,MP,Starts"
```

Bài 4: Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024 từ trang web <https://www.footballtransfers.com>.

```
1 def get_player_data(url):
2     driver.get(url)
3     time.sleep(5) # Đợi trang tải hoàn tất
4
5     # Tìm phần tử <tbody> chứa thông tin cầu thủ
6     table_body = driver.find_element(By.ID, 'player-table-body')
7     rows = table_body.find_elements(By.TAG_NAME, 'tr')
8
9     # Tạo danh sách để lưu thông tin cầu thủ
10    player_list = []
11
12    # Duyệt qua từng hàng trong bảng
13    for row in rows:
14        # Tạo một danh sách để lưu dữ liệu của từng ô trong hàng
15        row_data = []
16
17        # Lấy tên cầu thủ
18        name_tag = row.find_element(By.CLASS_NAME, 'td-player').find_element(By.TAG_NAME, 'a')
19        name = name_tag.text.strip() if name_tag else "N/A"
20        row_data.append(name)
21
```

1, Hàm `get_player_data(url)`:

Hàm này mở một URL bằng Selenium và đợi 5 giây để đảm bảo trang tải hoàn tất.

Sử dụng Selenium để tìm và lấy dữ liệu từ các hàng (<tr>) trong phần tử <tbody> có ID 'player-table-body'.

Lấy thông tin về từng cầu thủ bao gồm tên, câu lạc bộ chuyển đi, câu lạc bộ chuyển đến, ngày chuyển nhượng và phí chuyển nhượng bằng cách xác định vị trí của các lớp (class) thích hợp trong mỗi hàng.

Dữ liệu của từng cầu thủ được lưu thành một danh sách con và thêm vào danh sách `player_list`, là đầu ra của hàm này.

```
1 driver = webdriver.Chrome()
2
3 urls = [...]
4
5 all_players = []
6
7
8 for url in urls:
9     players = get_player_data(url)
10    all_players.extend(players)
```

## 2, Khởi tạo webdriver và chạy hàm cho từng URL:

`webdriver.Chrome()` khởi tạo một phiên làm việc với Chrome.

Danh sách `urls` chứa các URL từ các trang khác nhau của trang web chuyển nhượng.

Vòng lặp duyệt qua từng URL, gọi hàm `get_player_data` và thêm dữ liệu cầu thủ thu thập được vào danh sách `all_players`.

```
1 driver.quit()
2
3 # Chuyển danh sách thành DataFrame và hiển thị kết quả
4 df = pd.DataFrame(all_players, columns=['Tên cầu thủ', 'Câu lạc bộ chuyển đi', 'Câu lạc bộ chuyển đến', 'Ngày chuyển nhượng', 'Phí chuyển nhượng'])
5 print(df)
```

## 3, Đóng trình duyệt và lưu dữ liệu vào DataFrame:

Sau khi thu thập tất cả dữ liệu, mã đóng trình duyệt bằng `driver.quit()`.

Dữ liệu từ `all_players` được chuyển vào một DataFrame của Pandas, và sau đó in ra để xem kết quả.