

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



**BÁO CÁO BÀI TẬP LỚN**  
**LẬP TRÌNH VỚI PYTHON**

**Đề tài: Thu thập và phân tích dữ liệu cầu thủ bóng đá**

**Sinh viên thực hiện:**

<b>Họ và tên</b>	Đặng Hữu Nghĩa
<b>Mã sinh viên</b>	B22DCCN601
<b>Nhóm lớp</b>	11

*HÀ NỘI, THÁNG 10/2024*

# MỤC LỤC

Câu 1 .....	2
Câu 2 .....	7
Câu 3 .....	12
Câu 4 .....	17

**Câu 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.*
- *Ghi kết quả ra file 'results.csv'*

### 1. Thư viện sử dụng:

- import pandas as pd: Nhập thư viện pandas, dùng để xử lý và phân tích dữ liệu.
- import requests: Nhập thư viện requests, dùng để gửi yêu cầu HTTP.
- from bs4 import BeautifulSoup: Nhập BeautifulSoup từ thư viện bs4, dùng để phân tích cú pháp HTML.

```
1 from functools import reduce
2 import pandas as pd
3 import requests
4 from bs4 import BeautifulSoup
5
```

### 2. Lấy dữ liệu ban đầu:

- Khai báo URL của trang web chứa thống kê Premier League.
- Gửi yêu cầu GET đến URL và lưu phản hồi vào biến r.
- Phân tích cú pháp nội dung HTML của phản hồi và lưu vào biến soup.

```
6
7 if __name__ == "__main__":
8     url = 'https://fbref.com/en/comps/9/2023-2024/2023-2024-Premier-League-Stats'
9     r = requests.get(url)
10    soup = BeautifulSoup(r.content, features='html.parser')
11
```

### 3. Lấy danh sách đội bóng:

- Tìm bảng chứa thống kê của Premier League
- Sử dụng hàm find để tìm kiếm các đường link của các đội bóng trong tag tbody
- Tìm thẻ <a> trong hàng.
- Kiểm tra xem thẻ <a> có tồn tại và chứa từ "squads" trong URL.

```

11
12     table = soup.find( name: 'table', attrs: {
13         'class': 'stats_table sortable min_width force_mobilize',
14         'id': 'results2023-202491_overall'
15     })
16     # Danh sách chứa các đội bóng và url đến đội bóng đó
17     teams_data = []
18     # Tìm các thẻ <a> trong <tbody> của table
19     tbody = table.find('tbody')
20     teams = tbody.find_all( name: 'a', href=True)
21

```

- Thêm tên đội bóng và URL vào danh sách

```

22     # Đưa dữ liệu về tên và link đội bóng vào danh sách
23     for team in teams:
24         if "squads" in team['href']:
25             team_name = team.text.strip()
26             team_url = "https://fbref.com" + team['href']
27             teams_data.append([team_name, team_url])
28

```

#### 4. Tạo danh sách dữ liệu cần thu thập:

- Khởi tạo biến kiểm tra tiêu đề bảng, có thể sử dụng để xác định khi nào cần lấy tiêu đề.
- Khởi tạo danh sách rỗng cho từng loại thống kê cầu thủ.

```

30
31     ok = 1 # Tạo 1 biến check header
32     # Tạo danh sách chứa dữ liệu của các hàng
33     base_data = []
34     goalkeep_data = []
35     shooting_data = []
36     passing_data = []
37     passtype_data = []
38     goalshot_data = []
39     defensive_data = []
40     possess_data = []
41     playtime_data = []
42     miscell_data = []

```

## 5. Xử lý dữ liệu cho từng danh sách:

- Lập qua danh sách đội bóng
- Gửi yêu cầu HTTP và phân tích dữ liệu
- Tìm bảng dữ liệu
- Lấy tên các cột
- Lấy dữ liệu từ các hàng của bảng
- Làm tương tự với các danh sách khác

```
43     # Xử lý dữ liệu cho từng mục
44     for team in teams_data:
45         print(f"dang xu li du lieu doi {team[0]}..")
46         r = requests.get(team[1])
47         soup = BeautifulSoup(r.content, features='html.parser')
48
49         # Tìm bảng chứa dữ liệu
50         table = soup.find( name='table', attrs={
51             'class': 'stats_table sortable min_width',
52             'id': 'stats_standard_9'
53         })
54         # Tìm tên các cột
55         if ok == 1:
56             b = []
57             b.append("Team")
58             tthead = table.find('thead').find_all('tr')
59             for x in tthead[1]:
60                 if x.text.strip() != "":
61                     b.append(x.text.strip())
62             base_data.append(b)
63         # Tìm các dữ liệu còn lại của bảng
64         tbody = table.find('tbody').find_all('tr')
65         for y in tbody:
66             tmp = []
67             tmp.append(team[0])
68             for x in y:
69                 if (x.text.strip() == ""):
70                     tmp.append("N/a")
71                 else:
72                     tmp.append(x.text.strip())
73             base_data.append(tmp)
```

## 6. Xử lý dữ liệu thu được

- Chuyển đổi danh sách dữ liệu thành DataFrame.
- Tạo 1 danh sách chứa tất cả các DataFrame của từng mục
- Loại bỏ những dữ liệu không cần thiết

```
286 # Tạo 1 list chứa tất cả các df
287 all_df = []
288 # Chuyển đổi list thành DataFrame và loại bỏ những hàng không cần thiết
289 base_data_df = pd.DataFrame(base_data[1:], columns=base_data[0])
290 base_data_df = base_data_df.drop(labels=['90s', '6ls', 'G+A', 'PK', 'npX6+xA6', 'Matches'], axis=1)
291 all_df.append(base_data_df.drop_duplicates(subset='Player'))
292
293 goalkeep_data_df = pd.DataFrame(goalkeep_data[1:], columns=goalkeep_data[0])
294 goalkeep_data_df = goalkeep_data_df.drop(labels=['Nation', 'Pos', 'Age', 'MP', 'Starts', 'Min', '90s', 'Matches'], axis=1)
295 all_df.append(goalkeep_data_df.drop_duplicates(subset='Player'))
296
297 shooting_data_df = pd.DataFrame(shooting_data[1:], columns=shooting_data[0])
298 shooting_data_df = shooting_data_df.drop(labels=['Nation', 'Pos', 'Age', '90s', 'Matches'], axis=1)
299 all_df.append(shooting_data_df.drop_duplicates(subset='Player'))
300
301 passing_data_df = pd.DataFrame(passing_data[1:], columns=passing_data[0])
302 passing_data_df = passing_data_df.drop(labels=['Nation', 'Pos', 'Age', '90s', 'Matches'], axis=1)
303 all_df.append(passing_data_df.drop_duplicates(subset='Player'))
304
305 passtype_data_df = pd.DataFrame(passtype_data[1:], columns=passtype_data[0])
306 passtype_data_df = passtype_data_df.drop(labels=['Nation', 'Pos', 'Age', '90s', 'Att', 'Matches'], axis=1)
307 all_df.append(passtype_data_df.drop_duplicates(subset='Player'))
308
309 goalshot_data_df = pd.DataFrame(goalshot_data[1:], columns=goalshot_data[0])
310 goalshot_data_df = goalshot_data_df.drop(labels=['Nation', 'Pos', 'Age', '90s', 'Matches'], axis=1)
311 all_df.append(goalshot_data_df.drop_duplicates(subset='Player'))
312
313 defensive_data_df = pd.DataFrame(defensive_data[1:], columns=defensive_data[0])
314 defensive_data_df = defensive_data_df.drop(labels=['Nation', 'Pos', 'Age', '90s', 'Matches'], axis=1)
315 all_df.append(defensive_data_df.drop_duplicates(subset='Player'))
316
317 possess_data_df = pd.DataFrame(possess_data[1:], columns=possess_data[0])
```

- Gộp các DataFrame lại

```
334 result = reduce(lambda left, right: pd.merge(left, right, on=['Player'], how='outer'), all_df)
335 result = result.drop_duplicates()
336 result = result.groupby('Player').first().reset_index()
337 result = result.T.drop_duplicates().T
```

- Lọc dữ liệu theo điều kiện và ghi vào file “Results.csv”

```
339 # Lọc theo điều kiện
340 def ch(a: str): 1 usage
341     if a == 'N/a':
342         return 0
343     return int(a.replace(_old: ',', _new: ''))
344
345
346 a = result['Min'].to_list()
347 b = [x for x in a if ch(x) > 90]
348 result = result[result['Min'].isin(b)].reset_index(drop=True)
349 print(result)
350 # Ghi thông tin vào file csv
351 result.to_csv("results.csv", index=False)
```

## 7. Kết quả:

part1.py × results.csv × part2.py × results2.csv × part3-KMeans, PCA.py × radarChartPlot.py

	Player (1)	Team (2)	Nation ...	Pos ...	A...	...	Starts_x ...	Min ...	Ast...	G-...	PKatt...	Cr...	Cr...	xG...	npG...	xAG...	PrgC...	PrgP...
1	Player	Team	Nation	Pos	Age	MP	Starts_x	Min	Ast_x	G-PK	PKatt_x	CrDY	CrDR	xG_x	npG_x	xAG_x	PrgC_x	PrgP_x
2	Aaron Cresswell	West Ham	eng ENG	DF,FW	33	11	4	436	0	0	0	1	0	0.0	0.0	0.4	4	26
3	Aaron Hickey	Brentford	sct SCO	DF	21	9	9	713	0	0	0	5	0	0.2	0.2	0.1	9	21
4	Aaron Ramsdale	Arsenal	eng ENG	GK	25	6	6	540	0	0	0	0	0	0.0	0.0	0.0	0	2
5	Aaron Ramsey	Burnley	eng ENG	MF,FW	20	14	5	527	0	0	0	1	0	0.3	0.3	0.4	8	8
6	Aaron Wan-Bissaka	Manchester Utd	eng ENG	DF	25	22	20	1,780	2	0	0	4	0	0.1	0.1	1.5	30	77
7	Abdoulaye Doucouré	Everton	ml MLI	FW,MF	30	32	32	2,629	1	7	0	7	0	8.8	8.8	2.9	55	97
8	Adam Lallana	Brighton	eng ENG	MF,FW	35	25	13	850	1	0	0	2	0	0.8	0.8	1.7	9	39
9	Adam Smith	Bournemouth	eng ENG	DF	32	28	25	2,150	2	0	0	6	0	0.1	0.1	1.3	24	89
10	Adam Webster	Brighton	eng ENG	DF	28	15	13	1,144	0	0	0	2	0	0.4	0.4	0.1	16	80
11	Adam Wharton	Crystal Palace	eng ENG	MF	19	16	15	1,297	3	0	0	2	0	0.3	0.3	2.4	14	79
12	Adama Traoré	Fulham	es ESP	FW,MF	27	17	1	377	3	2	0	2	0	1.5	1.5	0.7	29	16
13	Albert Sambi Lokonga	Luton Town	be BEL	MF	23	17	16	1,303	3	1	0	4	0	0.6	0.6	1.4	29	72
	Alejandro Garnacho	Manchester Utd	ar ARG	FW	19	36	30	2,565	4	7	0	4	0	8.4	8.3	5.1	178	62

TextTable Editor

Text Table Editor

**Câu 2:** - *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*

- *Vẽ histogram 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*

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

- Lấy cột số: Xác định các cột trong DataFrame có kiểu dữ liệu số.
- Hàm con values: Sắp xếp DataFrame theo từng cột, lấy tên 3 cầu thủ đứng đầu và 3 cầu thủ đứng cuối.
- Tạo DataFrame kết quả: Lưu kết quả vào một DataFrame mới.
- In kết quả: Hiển thị tên 3 cầu thủ đứng đầu và đứng cuối cho mỗi chỉ số.

```
def get_top_player(df): 1 usage
10     # Tìm kiếm các cột kiểu số và đưa vào danh sách
11     numeric_col = df.select_dtypes(include=['number'])
12     numeric_col.columns.tolist()
13
14     def values(column:str):
15         df_sorted = df.sort_values(by=column)
16         a=list(df_sorted['Player'].head(3).values)
17         b=list(df_sorted['Player'].tail(3).values)
18         return (a+b)
19
20     result_top=pd.DataFrame()
21     for x in numeric_col:
22         result_top[x]=values(x)
23     print(result_top)
```



### Chạy thử:

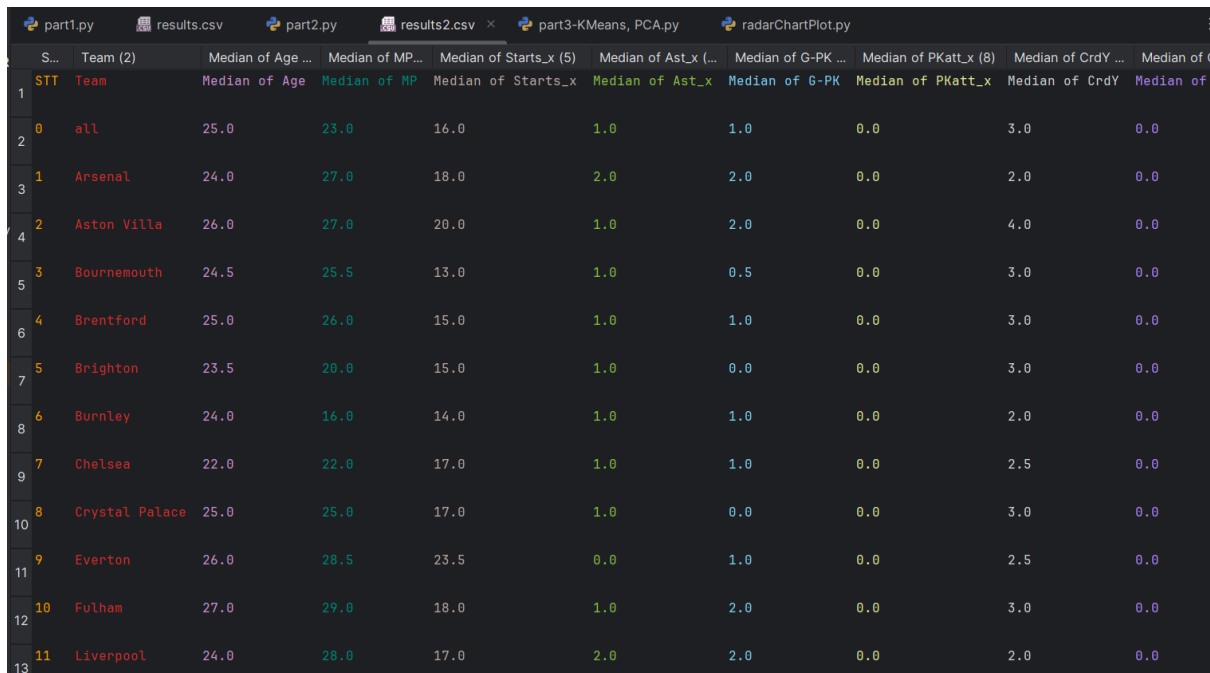
	Age	...	Lost_y
0	Lewis Miley	...	Łukasz Fabiański
1	Leon Chiwome	...	Aaron Ramsdale
2	Wilson Odobert	...	Đorđe Petrović
3	Thiago Silva	...	Dominic Solanke
4	Ashley Young	...	Dominic Calvert-Lewin
5	Łukasz Fabiański	...	Carlton Morris

## 2. Tìm trung vị, trung bình và độ lệch chuẩn

- Tìm cột số: Xác định các cột có kiểu dữ liệu số.
- Tính thống kê tổng thể: Tính trung vị, trung bình và độ lệch chuẩn cho tất cả các chỉ số, làm tròn đến 2 chữ số thập phân.
- Tạo bảng tổng thể: Tạo DataFrame chứa các thống kê tổng thể cho tất cả các đội.
- Tính thống kê theo đội: Tính trung vị, trung bình và độ lệch chuẩn cho từng đội.
- Tạo bảng theo đội: Tạo DataFrame chứa các thống kê theo từng đội.
- Gộp và lưu trữ: Gộp hai bảng và ghi kết quả vào file results2.csv, sau đó in nội dung file.

```
def get_statistics(df):  
    numeric_col = df.select_dtypes(include=['number'])  
    numeric_columns_list = numeric_col.columns.tolist()  
    median_all = numeric_col.median().round(2)  
    mean_all = numeric_col.mean().round(2)  
    std_all = numeric_col.std().round(2)  
    overall_df = pd.DataFrame({  
        'STT': [0],  
        'Team': ['all'],  
        **{f'Median of {col}': [median_all[col]] for col in numeric_col},  
        **{f'Mean of {col}': [mean_all[col]] for col in numeric_col},  
        **{f'Std of {col}': [std_all[col]] for col in numeric_col}  
    })  
    median_team = df.groupby('Team')[numeric_columns_list].median().round(2)  
    mean_team = df.groupby('Team')[numeric_columns_list].mean().round(2)  
    std_team = df.groupby('Team')[numeric_columns_list].std().round(2)  
    #Gộp các bảng này thành 1 bảng  
    team_df = pd.DataFrame({  
        'STT': range(1, len(median_team) + 1),  
        'Team': median_team.index,  
        **{f'Median of {col}': median_team[col].values for col in numeric_col},  
        **{f'Mean of {col}': mean_team[col].values for col in numeric_col},  
        **{f'Std of {col}': std_team[col].values for col in numeric_col}  
    })  
    final_df = pd.concat([overall_df, team_df], ignore_index=True)  
    final_df.to_csv('path_or_buf': 'results2.csv', index=False)  
    print(pd.read_csv('results2.csv'))
```

## Chạy thử:



The screenshot shows a Jupyter Notebook with several tabs: 'part1.py', 'results.csv', 'part2.py', 'results2.csv', 'part3-KMeans, PCA.py', and 'radarChartPlot.py'. The active tab is 'results2.csv', which displays a DataFrame with 13 rows and 10 columns. The columns are: 'STT', 'Team', 'Median of Age', 'Median of MP', 'Median of Starts\_x (5)', 'Median of Ast\_x', 'Median of G-PK', 'Median of PKatt\_x (8)', 'Median of CrdY', and 'Median of ...'. The rows represent different football teams, with the first row being 'all' and the subsequent rows being individual teams like Arsenal, Aston Villa, Bournemouth, etc.

STT	Team	Median of Age	Median of MP	Median of Starts_x (5)	Median of Ast_x	Median of G-PK	Median of PKatt_x (8)	Median of CrdY	Median of ...
0	all	25.0	23.0	16.0	1.0	1.0	0.0	3.0	0.0
1	Arsenal	24.0	27.0	18.0	2.0	2.0	0.0	2.0	0.0
2	Aston Villa	26.0	27.0	20.0	1.0	2.0	0.0	4.0	0.0
3	Bournemouth	24.5	25.5	13.0	1.0	0.5	0.0	3.0	0.0
4	Brentford	25.0	26.0	15.0	1.0	1.0	0.0	3.0	0.0
5	Brighton	23.5	20.0	15.0	1.0	0.0	0.0	3.0	0.0
6	Burnley	24.0	16.0	14.0	1.0	1.0	0.0	2.0	0.0
7	Chelsea	22.0	22.0	17.0	1.0	1.0	0.0	2.5	0.0
8	Crystal Palace	25.0	25.0	17.0	1.0	0.0	0.0	3.0	0.0
9	Everton	26.0	28.5	23.5	0.0	1.0	0.0	2.5	0.0
10	Fulham	27.0	29.0	18.0	1.0	2.0	0.0	3.0	0.0
11	Liverpool	24.0	28.0	17.0	2.0	2.0	0.0	2.0	0.0

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

- Tạo thư mục lưu trữ (nếu chưa có):
  - Tạo thư mục histograms\_all để lưu biểu đồ toàn giải.
  - Tạo thư mục histograms\_teams để lưu biểu đồ cho từng đội.
- Vẽ biểu đồ cho toàn giải:
  - Duyệt qua từng cột trong DataFrame và vẽ histogram.
  - Lưu các biểu đồ vào thư mục histograms\_all.
- Vẽ biểu đồ cho từng đội:
  - Lấy danh sách các đội và tạo thư mục riêng cho mỗi đội.
  - Duyệt qua các đội, lọc dữ liệu và vẽ histogram cho mỗi cột.
  - Lưu các biểu đồ vào thư mục tương ứng của đội.
- In thông báo hoàn thành:
  - In ra thông báo khi hoàn tất việc vẽ biểu đồ cho toàn giải và các đội.

## Vẽ toàn giải:

```
59 def print_histogram(df): 1usage
60     # Tên thư mục để lưu trữ các biểu đồ toàn giải
61     output_folder_1 = "histograms_all"
62
63     # Tạo thư mục nếu chưa tồn tại
64     if not os.path.exists(output_folder_1):
65         os.makedirs(output_folder_1)
66
67     # Vẽ histogram cho toàn giải
68     for col in df:
69         plt.figure(figsize=(8, 6))
70         sns.histplot(df[col], bins=20, kde=True, color='blue')
71         plt.title(f'Histogram of {col} - Toàn Giải')
72         plt.xlabel(col)
73         plt.ylabel('Số lượng cầu thủ ')
74         plt.grid(visible=True, linestyle='--', alpha=0.5)
75         # Lưu biểu đồ vào thư mục "histograms_all"
76         plt.savefig(os.path.join(output_folder_1, f"{df.columns.get_loc(col)}.png"))
77         plt.close()
78
79     print("Đã vẽ xong biểu đồ cho toàn giải")
```

## Vẽ các đội:

```
81     # Tên thư mục để lưu trữ các biểu đồ các đội
82     output_folder_2 = "histograms_teams"
83
84     # Tạo thư mục nếu chưa tồn tại
85     if not os.path.exists(output_folder_2):
86         os.makedirs(output_folder_2)
87
88     # Vẽ histogram cho từng đội
89     teams = df['Team'].unique()
90     for team in teams:
91         # Tên thư mục của đội
92         team_folder = os.path.join(output_folder_2, team)
93         # Tạo thư mục nếu chưa tồn tại
94         if not os.path.exists(team_folder):
95             os.makedirs(team_folder)
96         team_data = df[df['Team'] == team]
97         for col in df:
98             plt.figure(figsize=(8, 6))
99             sns.histplot(team_data[col], bins=20, kde=True, color='green')
100             plt.title(f'Histogram of {col} - {team}')
101             plt.xlabel(col)
102             plt.ylabel('Số lượng cầu thủ')
103             plt.grid(visible=True, linestyle='--', alpha=0.5)
104             # Lưu biểu đồ vào thư mục của đội
105             plt.savefig(os.path.join(team_folder, f"{df.columns.get_loc(col)}.png"))
106             plt.close()
107
108     print(f"Đã vẽ xong biểu đồ cho đội {team}")
```

#### 4. Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải

- Chuẩn bị dữ liệu: Lấy danh sách các cột số từ DataFrame df.
- Tính trung bình: Nhóm dữ liệu theo từng đội và tính giá trị trung bình cho mỗi chỉ số.
- Xác định đội xuất sắc: Duyệt qua từng chỉ số để tìm đội có giá trị trung bình cao nhất và lưu trữ kết quả.
- Tạo DataFrame: Chuyển đổi kết quả thành DataFrame và in ra.
- Đếm tần suất: Đếm số lần mỗi đội xuất hiện trong danh sách đội tốt nhất và sắp xếp theo tần suất.
- In kết quả: In ra tên đội có số điểm cao nhất và tần suất xuất hiện của nó.

```
109 def get_best_team(df): 1 usage
110     #chuan bi
111     results=[]
112     numeric_columns = df.select_dtypes(include=['number'])
113     numeric_columns_list = numeric_columns.columns.tolist()
114     mean_team = df.groupby('Team')[numeric_columns_list].mean().round(2)
115     for x in numeric_columns_list :
116         team=mean_team[x].idxmax()
117         value=mean_team[x].max()
118         results.append([team,x,value])
119     df_results = pd.DataFrame(results,columns=["Teams","Status","Value"])
120     print(df_results)
121     # Đếm tần suất của từng đội
122     team_counts = Counter([row[0] for row in results])
123     # Chuyển kết quả đếm tần suất thành dạng bảng và sắp xếp nó
124     frequency_table = [[team, count] for team, count in team_counts.items()]
125     frequency_table.sort(key=lambda x: x[1], reverse=True)
126     #In ra Teams có điểm số cao nhất
127     print(frequency_table[0][0],frequency_table[0][1])
```

**Chạy thử:**

```
      Teams  Status  Value
0    West Ham    Age  28.33
1    Fulham     MP   27.24
2    Fulham  Starts_x  19.90
3  Manchester City  Ast_x   3.24
4    Manchester City   G-PK   4.05
..      ...      ...    ...
143  Liverpool    Off_y   4.55
144  Sheffield Utd    OG    0.25
145  Liverpool    Recov  92.00
146  Everton     Won   30.73
147  West Ham    Lost_y  26.76

[148 rows x 3 columns]
Manchester City 49
```

➔ Manchester City là đội có phong độ tốt nhất giải

**Câu 3:** - 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

### 1. Sử dụng thuật toán K-means và PCA vẽ phân cụm các điểm dữ liệu

- Sử dụng các thư viện matplotlib, numpy, pandas, sklearn.decomposition.PCA, và sklearn.preprocessing.StandardScaler để xử lý và trực quan hóa dữ liệu.
- Hàm plot\_kmeans:
  - Vẽ biểu đồ phân cụm K-means.
  - Sử dụng colormap 'viridis' để phân biệt các cụm bằng màu sắc.
  - Vẽ các điểm dữ liệu và tâm của các cụm.

```
8 def plot_kmeans(data, centroids, clusters, step): 1 usage
9 plt.figure(figsize=(8, 6))
10
11 # Tạo màu sắc ngẫu nhiên cho các cụm
12 colors = plt.cm.get_cmap(name='viridis', k) # Sử dụng colormap 'viridis' với k màu
13
14 # Vẽ các điểm dữ liệu theo cụm
15 for i in range(k):
16     points = data[clusters == i]
17     plt.scatter(points[:, 0], points[:, 1], s=50, color=colors(i), label=f'Cluster {i}')
18     plt.scatter(centroids[i, 0], centroids[i, 1], s=200, color=colors(i), marker='X', edgecolor='k')
19
20 plt.title('K-means Clustering of Football Players')
21 plt.xlabel('PC1')
22 plt.ylabel('PC2')
23 plt.legend()
24 plt.show()
25
```

- Trong khối main
  - Đọc dữ liệu từ file CSV: Sử dụng pd.read\_csv() để tải dữ liệu từ results.csv.
  - Xử lý dữ liệu:
    - Loại bỏ các cột không phải số.
    - Thay thế giá trị NaN bằng giá trị trung bình của các cột.

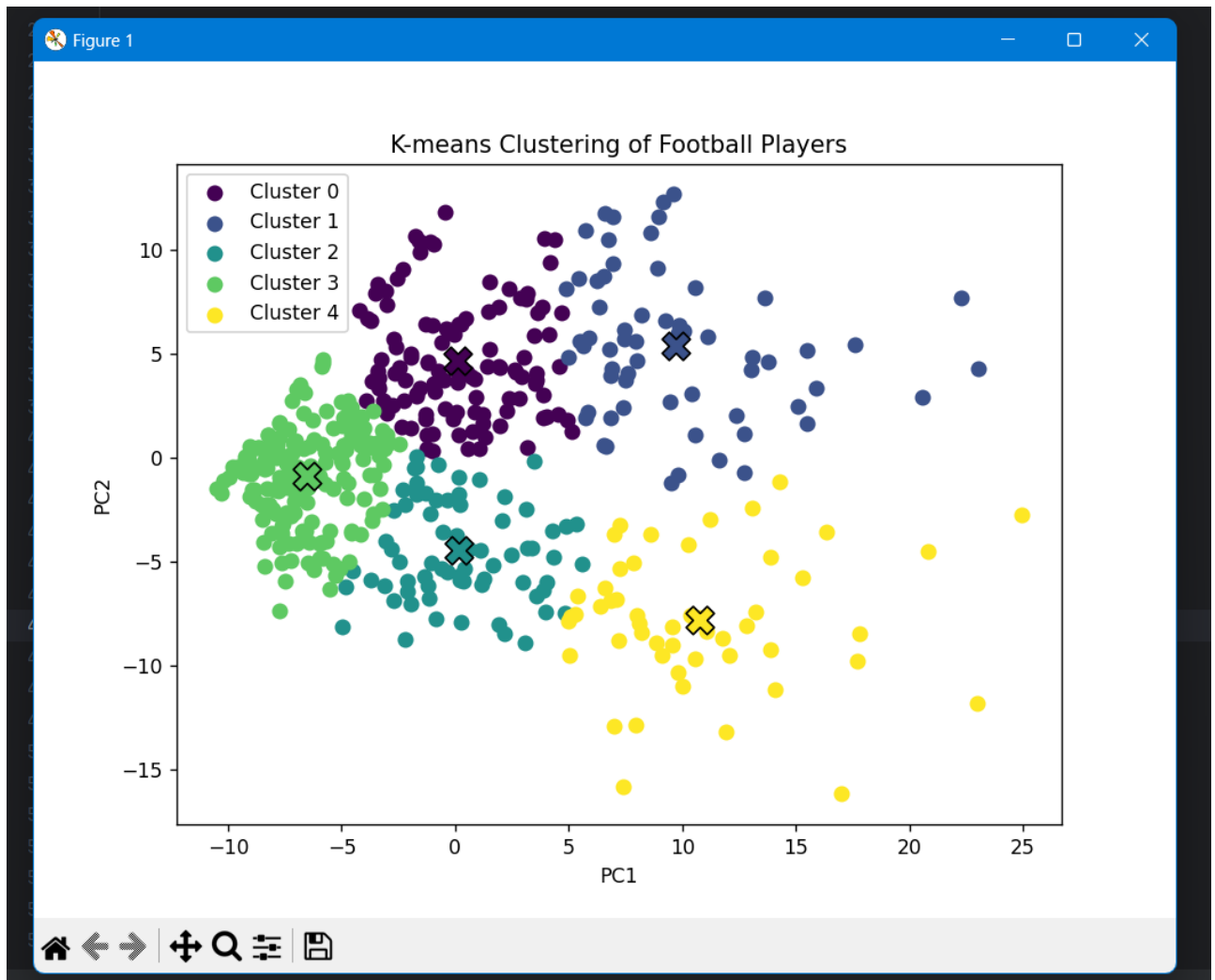
- Chuẩn hóa dữ liệu: Sử dụng StandardScaler để chuẩn hóa dữ liệu cho phù hợp.
- Áp dụng PCA: Giảm số chiều của dữ liệu xuống 2 để có thể trực quan hóa.
- Khởi tạo K-means:
  - Đặt số lượng cụm k (ở đây là 5).
  - Khởi tạo các tâm cụm ngẫu nhiên từ dữ liệu.
  - Khởi tạo nhãn cho các điểm dữ liệu (các cụm).
- Thuật toán K-means:
  - Lặp qua một số bước tối đa (ở đây là 100):
  - Gán nhãn cho từng điểm dữ liệu dựa trên khoảng cách đến các tâm cụm.
  - Cập nhật các tâm cụm bằng cách tính trung bình của các điểm dữ liệu trong cùng một cụm.
  - Kiểm tra xem các tâm cụm có thay đổi không; nếu không, vẽ biểu đồ và dừng lặp.

```

27 if __name__ == "__main__":
28     data = pd.read_csv('results.csv')
29     data = data.select_dtypes(exclude=['object'])
30     data = data.fillna(data.mean())
31     # Chuẩn hóa dữ liệu
32     scaler_standard = StandardScaler() # Khởi tạo
33     data = pd.DataFrame(scaler_standard.fit_transform(data), columns=data.columns)
34     # Áp dụng PCA giảm số chiều xuống 2
35     pca = PCA(n_components=2)
36     data = pca.fit_transform(data)
37     data = pd.DataFrame(data, columns=['PC1', 'PC2'])
38     k = 5
39     # Khởi tạo ngẫu nhiên các tâm cụm
40     centroids = data.sample(n=k).values
41     # Khởi tạo nhãn cho các điểm dữ liệu
42     clusters = np.zeros(data.shape[0])
43     epochs = 100
44     for step in range(epochs):
45         # Bước 1: Gán nhãn dựa trên khoảng cách đến các tâm cụm
46         for i in range(len(data)):
47             distances = np.linalg.norm(data.values[i] - centroids, axis=1)
48             clusters[i] = np.argmin(distances)
49         # Bước 2: Cập nhật các tâm cụm
50         new_centroids = np.array([data.values[clusters == j].mean(axis=0) for j in range(k)])
51         # Kiểm tra nếu các tâm cụm không thay đổi thì kết thúc
52         if np.all(centroids == new_centroids):
53             # Vẽ biểu đồ
54             plot_kmeans(data.values, centroids, clusters, step)
55             break
56         centroids = new_centroids

```

### Chạy thử:



### Phân tích:

- Nên phân loại cầu thủ làm 5 nhóm:
  - Cầu thủ tấn công/tiền đạo
  - Cầu thủ phòng ngự/thủ môn
  - Các tiền vệ tấn công/ makeplayer
  - Các tiền vệ phòng ngự
  - Các cầu thủ đa năng
- Chia như vậy vì:
  - có thể phù hợp với các vị trí trên sân
  - tạo sự cân bằng và kết quả phân cụm rõ ràng, tránh phức tạp

- Nhận xét kết quả:
  - Các nhóm có sự phân tách tốt, có ít sự chồng chéo giữa các nhóm
  - Hình dạng của các nhóm khác nhau, một số nhóm có hình dạng tương đối tròn, trong khi các nhóm khác có hình dạng kéo dài hơn
  - Phân bố không đều, một số nhóm có mật độ điểm cao hơn, trong khi các nhóm khác có các điểm phân tán hơn.

## 2. Vẽ biểu đồ radarchart so sánh cầu thủ

- **Đọc dữ liệu:** Đọc file CSV

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import numpy as np
4  import argparse
5
6  def load_data(file_path): 1 usage
7      return pd.read_csv(file_path)
8
```

- **Hàm vẽ biểu đồ radar (radar\_chart function):**

- Nhận dữ liệu data, tên của hai cầu thủ (player1 và player2), và danh sách các chỉ số (attributes) cần so sánh.

```
def radar_chart(data, player1, player2, attributes): 1 usage
    # Lọc dữ liệu cho từng cầu thủ
    p1_data = data[data['Player'] == player1]
    p2_data = data[data['Player'] == player2]
    # Kiểm tra nếu không tìm thấy cầu thủ
    if p1_data.empty or p2_data.empty:
        print("Không tìm thấy cầu thủ.")
```

- Lọc dữ liệu:
  - Sử dụng điều kiện để chọn dữ liệu của cầu thủ tương ứng.
  - Nếu không tìm thấy cầu thủ nào trong dữ liệu, chương trình sẽ báo lỗi và kết thúc.
- Lấy giá trị các chỉ số:
  - Các giá trị thuộc tính của mỗi cầu thủ được lấy từ DataFrame và chuyển thành một mảng.
  - Để khép kín biểu đồ radar, giá trị đầu tiên được thêm lại vào cuối mảng, tạo thành vòng tròn hoàn chỉnh.



```
# Lấy giá trị các thuộc tính
p1_values = p1_data[attributes].values.flatten()
p2_values = p2_data[attributes].values.flatten()
# Xây dựng góc của biểu đồ radar
num_vars = len(attributes)
angles = np.linspace(start=0, 2 * np.pi, num_vars, endpoint=False).tolist()
# Hoàn thành vòng radar
p1_values = np.concatenate((p1_values, [p1_values[0]]))
p2_values = np.concatenate((p2_values, [p2_values[0]]))
angles += angles[:1]
```

## – Vẽ biểu đồ

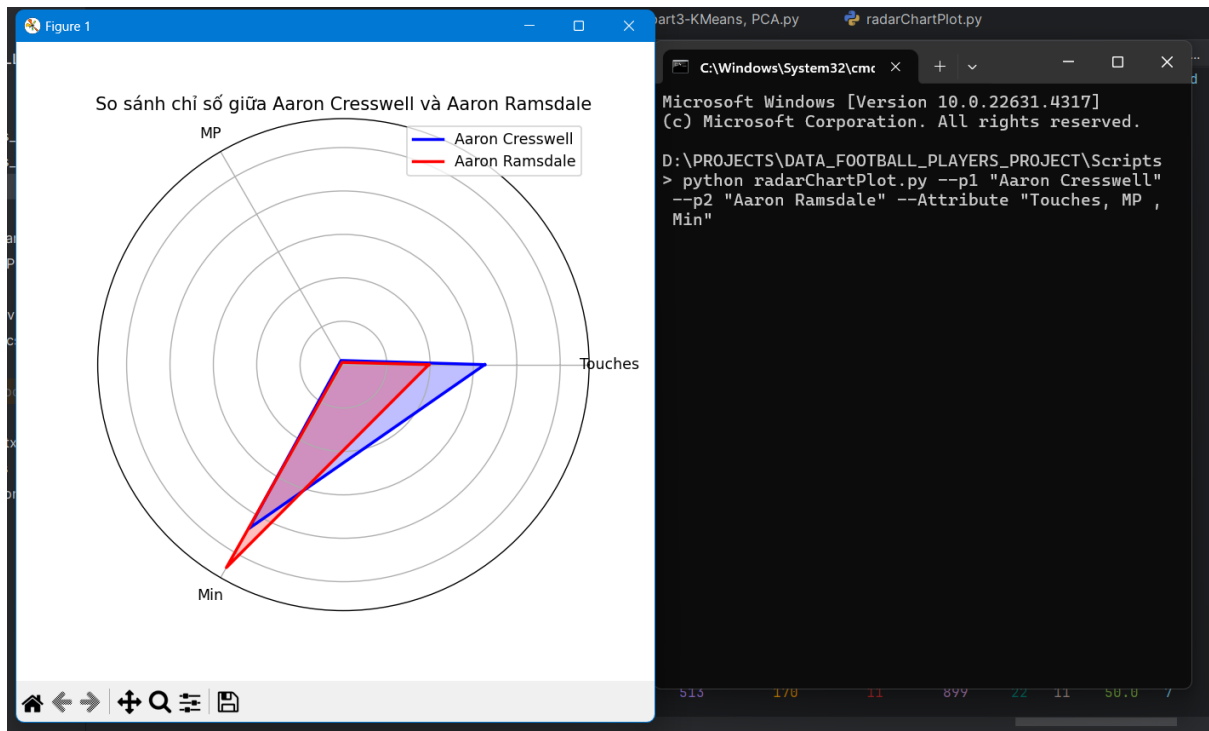
```
fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
ax.fill(*args: angles, p1_values, color='blue', alpha=0.25)
ax.fill(*args: angles, p2_values, color='red', alpha=0.25)
ax.plot(*args: angles, p1_values, color='blue', linewidth=2, label=player1)
ax.plot(*args: angles, p2_values, color='red', linewidth=2, label=player2)
# Cấu hình các nhãn thuộc tính
ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(attributes)
plt.title(f"So sánh chỉ số giữa {player1} và {player2}")
plt.legend(loc='upper right')
plt.show()
```

## • Chạy chương trình:

- Đọc file CSV chứa dữ liệu và gọi hàm radar\_chart với các tham số đã được chỉ định qua dòng lệnh.
- Biểu đồ radar hiển thị sự so sánh giữa hai cầu thủ trên các chỉ số, giúp người dùng dễ dàng thấy được sự khác biệt giữa họ.
- Chương trình cho phép người dùng nhập tên hai cầu thủ (--p1, --p2) và danh sách các chỉ số (--Attribute) qua dòng lệnh.
- Các chỉ số phải được nhập cách nhau bằng dấu phẩy. Ví dụ: "Goals, Assists, Passes".

```
40 if __name__ == "__main__":
41     parser = argparse.ArgumentParser(description="Vẽ biểu đồ radar so sánh cầu thủ")
42     parser.add_argument(*name_or_flags: "--p1", type=str, required=True, help="Tên cầu thủ thứ nhất")
43     parser.add_argument(*name_or_flags: "--p2", type=str, required=True, help="Tên cầu thủ thứ hai")
44     parser.add_argument(*name_or_flags: "--Attribute", type=str, required=True, help="Danh sách các chỉ số cần so sánh, cách nhau bởi dấu phẩy")
45     args = parser.parse_args()
46     attributes = [attr.strip() for attr in args.Attribute.split(",")]
47     # Đọc dữ liệu và vẽ biểu đồ
48     data = load_data("results.csv")
49     radar_chart(data, args.p1, args.p2, attributes)
```

## Chạy thử:



**Câu 4:** - 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.