**BÁO CÁO BÀI TẬP LỚN**

**HỌ VÀ TÊN: THÀO A BẢY.**

**MSV: B22DCCN072.**

**Nhóm: 11**

**CÂU 1:** **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.**

* **Code:**

import requests

from bs4 import BeautifulSoup as bs

import pandas as pd

# Danh sách các URL cho các đội tuyển

urls = [

'https://fbref.com/en/squads/822bd0ba/2023-2024/Liverpool-Stats',

'https://fbref.com/en/squads/b8fd03ef/2023-2024/Manchester-City-Stats',

'https://fbref.com/en/squads/18bb7c10/2023-2024/Arsenal-Stats',

'https://fbref.com/en/squads/8602292d/2023-2024/Aston-Villa-Stats',

'https://fbref.com/en/squads/4ba7cbea/2023-2024/Bournemouth-Stats',

'https://fbref.com/en/squads/cd051869/2023-2024/Brentford-Stats',

'https://fbref.com/en/squads/d07537b9/2023-2024/Brighton-and-Hove-Albion-Stats',

'https://fbref.com/en/squads/943e8050/2023-2024/Burnley-Stats',

'https://fbref.com/en/squads/cff3d9bb/2023-2024/Chelsea-Stats',

'https://fbref.com/en/squads/47c64c55/2023-2024/Crystal-Palace-Stats',

'https://fbref.com/en/squads/d3fd31cc/2023-2024/Everton-Stats',

'https://fbref.com/en/squads/fd962109/2023-2024/Fulham-Stats',

'https://fbref.com/en/squads/e297cd13/2023-2024/Luton-Town-Stats',

'https://fbref.com/en/squads/19538871/2023-2024/Manchester-United-Stats',

'https://fbref.com/en/squads/b2b47a98/2023-2024/Newcastle-United-Stats',

'https://fbref.com/en/squads/e4a775cb/2023-2024/Nottingham-Forest-Stats',

'https://fbref.com/en/squads/1df6b87e/2023-2024/Sheffield-United-Stats',

'https://fbref.com/en/squads/361ca564/2023-2024/Tottenham-Hotspur-Stats',

'https://fbref.com/en/squads/7c21e445/2023-2024/West-Ham-United-Stats',

'https://fbref.com/en/squads/8cec06e1/2023-2024/Wolverhampton-Wanderers-Stats'

]

Tong\_hop\_team = []

for url in urls:

# Gửi yêu cầu tới URL

r = requests.get(url)

soup = bs(r.text, 'html.parser')

tables = soup.find\_all('table')

if tables:

*#### Bảng cầu thủ*

bang = []

*# Lấy tên đội bóng*

team = tables[0].find('caption').find('span').text.strip()

tmp = team.replace(":", "").split(" ")

doi = ""

for i in range(1,len(tmp)-2):

doi += tmp[i]+" "

rows0 = tables[0].find('tbody').find\_all('tr')

for im in rows0:

name = im.find('th').text.strip()

cols = im.find\_all('td')

Play = []

*#### Nation*

Play.append(cols[0].text.strip())

*#### Team*

Play.append(doi.strip())

*#### Position, Age, Playing time*

for i in range(1, 6):

time = cols[i].text.strip()

Play.append(time)

*# Performance*

non\_Penalty = cols[10].text.strip()

Play.append(non\_Penalty)

Penalty\_Goalsy = cols[11].text.strip()

Play.append(non\_Penalty)

Assists = cols[8].text.strip()

Play.append(Assists)

Yellow\_Cards = cols[13].text.strip()

Play.append(Yellow\_Cards)

Red\_Cards = cols[14].text.strip()

Play.append(Red\_Cards)

*# Expected*

for i in range(15, 18):

tmp = cols[i].text.strip()

Play.append(tmp)

*# Progression: PrgC, PrgP, PrgR*

Pro = []

for i in range(19, 32):

tmp = cols[i].text.strip()

Pro.append(tmp)

*# Bảng tổng hợp thông tin*

bang.append([name] + Play + Pro)

df1 = pd.DataFrame(bang, columns=['Name', 'Nation', 'Team', 'Position', 'Age', 'matches\_played', 'starts', 'minutes',

'non\_Penalty\_Goals', 'Penalty Goals', 'Assists', 'Yellow\_Cards', 'Red\_Cards',

'xG', 'npxG', 'xAG', 'PrgC', 'PrgP', 'PrgR',

'Gls', 'Ast', 'G+A', 'G-PK', 'G+A-PK', 'xG\_per\_90', 'xAG\_per\_90',

'xG+xAG', 'npxG\_per\_90', 'npxG+xAG\_per\_90'])

*#### Bảng Goalkeeping*

bang2 = []

rows1 = tables[2].find('tbody').find\_all('tr')

for im in rows1:

name2 = im.find('th').text.strip()

cols1 = im.find\_all('td')

Goal = []

for i in range(7, 17):

tmp = cols1[i].text.strip()

Goal.append(tmp)

pen = []

for i in range(17, 22):

tmp = cols1[i].text.strip()

pen.append(tmp)

bang2.append([name2] + Goal + pen)

df2 = pd.DataFrame(bang2, columns=['Name', 'GA', 'GA90', 'SoTA', 'Saves', 'Save%', 'W', 'D', 'L', 'CS', 'CS%',

'PKatt', 'PKA', 'PKsv', 'PKm', 'Save%\_again'])

*#### Bảng Shooting*

bang3 = []

rows3 = tables[4].find('tbody').find\_all('tr')

for im in rows3:

name3 = im.find('th').text.strip()

cols3 = im.find\_all('td')

Sta = []

for i in range(4, 16):

tmp = cols3[i].text.strip()

Sta.append(tmp)

Exp = []

for i in range(16, 21):

tmp = cols3[i].text.strip()

Exp.append(tmp)

bang3.append([name3] + Sta + Exp)

df3 = pd.DataFrame(bang3, columns=['Name', 'Gls', 'Sh', 'SoT', 'SoT%', 'Sh/90', 'SoT/90', 'G/Sh', 'G/SoT',

'Dist', 'FK', 'PK', 'PKatt', 'xG', 'npxG', 'npxG/Sh', 'G-xG', 'np:G-xG'])

*#### Bảng Passing*

bang4 = []

rows4 = tables[5].find('tbody').find\_all('tr')

for im in rows4:

name4 = im.find('th').text.strip()

cols4 = im.find\_all('td')

bang\_phu = []

# Total: Cmp, Att, Cmp%, TotDist, PrgDist

for i in range(4, 9):

tmp = cols4[i].text.strip()

bang\_phu.append(tmp)

# Short: Cmp, Att, Cmp%

for i in range(9,12):

tmp = cols4[i].text.strip()

bang\_phu.append(tmp)

# Medium: Cmp, Att, Cmp%

for i in range(12,15):

tmp = cols4[i].text.strip()

bang\_phu.append(tmp)

# Long: Cmp, Att, Cmp%

for i in range(15,18):

tmp = cols4[i].text.strip()

bang\_phu.append(tmp)

# Expected: Ast, xAG, xA, A-xAG, KP, 1/3, PPA, CrsPA, PrgP

for i in range(18, 27):

tmp = cols4[i].text.strip()

bang\_phu.append(tmp)

bang4.append([name4] + bang\_phu)

df4 = pd.DataFrame(bang4, columns =['Name', 'Total Cmp', 'Total Att', 'Total Cmp%', 'TotDist', 'PrgDist',

'Short Cmp', 'Short Att', 'Short Cmp%',

'Medium Cmp', 'Medium Att', 'Medium Cmp%',

'Long Cmp', 'Long Att', 'Long Cmp%',

'Ast', 'xAG', 'xA', 'A-xAG', 'Key Passes', 'Final Third Passes','PPA', 'CrsPA', 'PrgP'])

*#### Bảng Pass Types*

bang5 = []

rows5 = tables[6].find('tbody').find\_all('tr')

for row in rows5:

name5 = row.find('th').text.strip()

cols5 = row.find\_all('td')

pass\_types = []

# Pass Types: Live, Dead, FK, TB, Sw, Crs, TI, CK

for i in range(5, 13):

pass\_types.append(cols5[i].text.strip())

# Corner Kicks: In, Out, Str

for i in range(13, 16):

pass\_types.append(cols5[i].text.strip())

# Outcomes: Cmp, Off, Blocks

for i in range(16, 19):

pass\_types.append(cols5[i].text.strip())

bang5.append([name5] + pass\_types)

df5 = pd.DataFrame(bang5, columns=[ 'Name',

'Pass Types.Live', 'Pass Types.Dead', 'Pass Types.FK', 'Pass Types.TB', 'Pass Types.Sw',

'Pass Types.Crs', 'Pass Types.TI', 'Pass Types.CK',

'Corner Kicks.In', 'Corner Kicks.Out', 'Corner Kicks.Str',

'Outcomes.Cmp', 'Outcomes.Off', 'Outcomes.Blocks' ])

*#### Bảng Goal and Shot Creation*

bang6 = []

rows6 = tables[7].find('tbody').find\_all('tr')

for row in rows6:

name6 = row.find('th').text.strip()

cols6 = row.find\_all('td')

nap = []

for i in range(4, 20):

nap.append(cols6[i].text.strip())

bang6.append([name6] + nap)

df6= pd.DataFrame(bang6, columns=[

'Name', 'SCA', 'SCA90', 'PassLive', 'PassDead', 'TO', 'Sh', 'Fld', 'Def',

'GCA', 'GCA90', 'GCA PassLive', 'GCA PassDead', 'GCA TO', 'GCA Sh', 'GCA Fld', 'GCA Def'

])

*#### Bảng Defensive Actions:*

bang7 = []

rows7 = tables[8].find('tbody').find\_all('tr')

for row in rows7:

name7 = row.find('th').text.strip()

cols7 = row.find\_all('td')

nap = []

for i in range(4, 20):

nap.append(cols7[i].text.strip())

bang6.append([name7] + nap)

df7= pd.DataFrame(bang7, columns=[

'Name', 'Tkl', 'TklW', 'Def 3rd', 'Mid 3rd', 'Att 3rd',

'Tkl Att', 'Tkl%', 'Tkl Lost', 'Blocks', 'Sh', 'Pass', 'Int', 'Tkl+Int', 'Clr', 'Err'

])

*#### Bảng Possession*

bang8 = []

rows8 = tables[9].find('tbody').find\_all('tr')

for row in rows8:

name8 = row.find('th').text.strip()

cols8= row.find\_all('td')

nap = []

for i in range(4, 26):

nap.append(cols8[i].text.strip())

bang8.append([name8] + nap)

df8= pd.DataFrame(bang8, columns=[

'Name', 'Touches', 'Def Pen', 'Def 3rd', 'Mid 3rd', 'Att 3rd', 'Att Pen', 'Live',

'Att', 'Succ', 'Succ%', 'Tkld', 'Tkld% ',

'Carries', 'TotDist', 'ProDist', 'ProgC', '1/3', 'CPA', 'Mis', 'Dis', 'Rec', 'PrgR'])

*#### Bảng Playing Time*

bang9 = []

rows9 = tables[10].find('tbody').find\_all('tr')

for row in rows9:

name9 = row.find('th').text.strip()

cols9= row.find\_all('td')

nap = []

for i in range(9, 18):

nap.append(cols9[i].text.strip())

for i in range(21, 23):

nap.append(cols9[i].text.strip())

bang9.append([name9] + nap)

df9= pd.DataFrame(bang9, columns=[

'Name', 'Starts', 'Mn/Start', 'Compl', 'Subs', 'Mn/Sub', 'unSub',

'PPM', 'onG', 'onGA', 'onxG', 'onxGA'

])

*#### Bảng Miscellaneous Stats*

bang10 = []

rows10= tables[11].find('tbody').find\_all('tr')

for row in rows10:

name10 = row.find('th').text.strip()

cols10= row.find\_all('td')

nap = []

for i in range(7, 11):

nap.append(cols10[i].text.strip())

for i in range(15, 20):

nap.append(cols10[i].text.strip())

bang10.append([name10] + nap)

df10= pd.DataFrame(bang10, columns=[

'Name', 'Fls', 'Fld', 'Off', 'Crs', 'OG', 'Recov',

'Aerial Won', 'Aerial Lost', 'Aerial Won%'

])

*#### Chuẩn hóa cột Tên về cùng 1 dạng chữ*

df1['Name'] = df1['Name'].str.strip().str.upper()

df2['Name'] = df2['Name'].str.strip().str.upper()

df3['Name'] = df3['Name'].str.strip().str.upper()

df4['Name'] = df4['Name'].str.strip().str.upper()

df5['Name'] = df5['Name'].str.strip().str.upper()

df6['Name'] = df6['Name'].str.strip().str.upper()

df7['Name'] = df7['Name'].str.strip().str.upper()

df8['Name'] = df8['Name'].str.strip().str.upper()

df9['Name'] = df9['Name'].str.strip().str.upper()

df10['Name'] = df10['Name'].str.strip().str.upper()

*#### Ghép các bảng dữ liệu dựa trên cột Tên*

merged\_df = df1

merged\_df = pd.merge(merged\_df, df2, on='Name', how='left')

merged\_df = pd.merge(merged\_df, df3, on='Name', how='left')

merged\_df = pd.merge(merged\_df, df4, on='Name', how='left')

merged\_df = pd.merge(merged\_df, df5, on='Name', how='left')

merged\_df = pd.merge(merged\_df, df6, on='Name', how='left')

merged\_df = pd.merge(merged\_df, df7, on='Name', how='left')

merged\_df = pd.merge(merged\_df, df8, on='Name', how='left')

merged\_df = pd.merge(merged\_df, df9, on='Name', how='left')

merged\_df = pd.merge(merged\_df, df10, on='Name', how='left')

*#### Chuyển cột minutes thành kiểu số nguyên*

merged\_df['minutes'] = pd.to\_numeric(merged\_df['minutes'].str.replace(',', ''), errors='coerce')

*#### Lọc cầu thủ có số phút thi nhiều hơn 90*

cau\_thu = merged\_df[merged\_df['minutes'] > 90]

*#### In ra tất cả các thuộc tính của các cầu thủ này*

Tong\_hop\_team.append(cau\_thu)

*#### Gộp tất cả các DataFrame vào một DataFrame duy nhất*

merged\_dfx = pd.concat(Tong\_hop\_team, ignore\_index=True)

*#### Sắp xếp tên cầu thủ theo tên và tuổi*

merged\_dfx = merged\_dfx.sort\_values(by=['Name', 'Age'], ascending=[True, False])

*#### Thay thế giá trị rỗng bằng 'N/a'*

merged\_dfx = merged\_dfx.fillna('N/a')

*#### Ghi kết quả ra file CSV*

merged\_dfx.to\_csv('results.csv', index=False)

*#### In ra DataFrame kết quả*

print(merged\_dfx)

**CÂU 2:**

1. **TÌM TOP 3 CẦU THỦ CÓ ĐIỂM CAO NHẤT VÀ THẤP NHẤT Ở MỖI CHỈSỐ.**

* **Code:**

import pandas as pd

*# Đọc dữ liệu từ tệp CSV*

data = pd.read\_csv('results.csv')

*# Danh sách các chỉ số*

attributes = [

'Team', 'Name',

'matches\_played', 'starts', 'minutes', 'non\_Penalty\_Goals', 'Penalty Goals', 'Assists',

'Yellow\_Cards', 'Red\_Cards', 'xG\_x', 'npxG\_x', 'xAG\_x', 'PrgC', 'PrgP\_x', 'PrgR\_x',

'Gls\_x', 'Ast\_x', 'G+A', 'G-PK', 'G+A-PK', 'xG\_per\_90', 'xAG\_per\_90', 'xG+xAG',

'npxG\_per\_90', 'npxG+xAG\_per\_90', 'Gls\_y', 'Sh\_x', 'SoT', 'SoT%', 'Sh/90', 'SoT/90',

'G/Sh', 'G/SoT', 'Dist', 'FK', 'PK', 'PKatt\_y', 'xG\_y', 'npxG\_y', 'npxG/Sh',

'G-xG', 'np:G-xG', 'Total Cmp', 'Total Att', 'Total Cmp%', 'TotDist\_x', 'PrgDist',

'Short Cmp', 'Short Att', 'Short Cmp%', 'Medium Cmp', 'Medium Att', 'Medium Cmp%',

'Long Cmp', 'Long Att', 'Long Cmp%', 'Ast\_y', 'xAG\_y', 'xA', 'A-xAG', 'Key Passes',

'Final Third Passes', 'PPA', 'CrsPA', 'PrgP\_y', 'Pass Types.Live', 'Pass Types.Dead',

'Pass Types.FK', 'Pass Types.TB', 'Pass Types.Sw', 'Pass Types.Crs', 'Pass Types.TI',

'Pass Types.CK', 'Corner Kicks.In', 'Corner Kicks.Out', 'Corner Kicks.Str',

'Outcomes.Cmp', 'Outcomes.Off', 'Outcomes.Blocks', 'SCA', 'SCA90', 'PassLive',

'PassDead', 'TO', 'Sh\_y', 'Fld\_x', 'Def', 'GCA', 'GCA90', 'GCA PassLive',

'GCA PassDead', 'GCA TO', 'GCA Sh', 'GCA Fld', 'GCA Def', 'Touches', 'Def Pen',

'Def 3rd\_y', 'Mid 3rd\_y', 'Att 3rd\_y', 'Att Pen', 'Live', 'Att', 'Succ', 'Succ%',

'Tkld', 'Tkld% ', 'Carries', 'TotDist\_y', 'ProDist', 'ProgC', '1/3', 'CPA', 'Mis',

'Dis', 'Rec', 'PrgR\_y', 'Starts', 'Mn/Start', 'Compl', 'Subs', 'Mn/Sub', 'unSub',

'PPM', 'onG', 'onGA', 'onxG', 'onxGA', 'Fls', 'Fld\_y', 'Off', 'Crs', 'OG',

'Recov', 'Aerial Won', 'Aerial Lost', 'Aerial Won%'

]

*# Chuyển đổi các cột số liệu sang kiểu số*

for attribute in attributes[2:]:

if attribute in data.columns:

data[attribute] = pd.to\_numeric(data[attribute], errors='coerce')

result = {

'Chỉ số': [],

'Top 3 cầu thủ có điểm cao nhất': [],

'Tên đội của cầu thủ có điểm cao nhất': [],

'Điểm cao nhất': [],

'Top 3 cầu thủ có điểm thấp nhất': [],

'Tên đội của cầu thủ có điểm thấp nhất': [],

'Điểm thấp nhất': []

}

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

for attribute in attributes:

if attribute in data.columns and pd.api.types.is\_numeric\_dtype(data[attribute]):

*# Lấy 3 cầu thủ có điểm cao nhất*

top\_3 = data.nlargest(3, attribute)[['Name', 'Team', attribute]].dropna()

*# Lấy 3 cầu thủ có điểm thấp nhất*

bottom\_3 = data.nsmallest(3, attribute)[['Name', 'Team', attribute]].dropna()

result['Chỉ số'].append(attribute)

result['Top 3 cầu thủ có điểm cao nhất'].append(", ".join(top\_3['Name']))

result['Tên đội của cầu thủ có điểm cao nhất'].append(", ".join(top\_3['Team']))

result['Điểm cao nhất'].append(", ".join(top\_3[attribute].astype(str)))

result['Top 3 cầu thủ có điểm thấp nhất'].append(", ".join(bottom\_3['Name']))

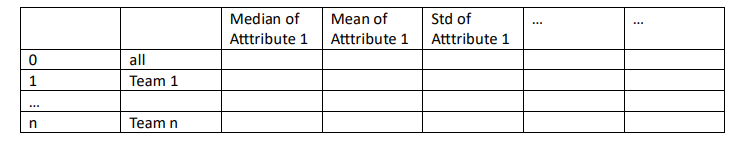
result['Tên đội của cầu thủ có điểm thấp nhất'].append(", ".join(bottom\_3['Team']))

result['Điểm thấp nhất'].append(", ".join(bottom\_3[attribute].astype(str)))

result\_df = pd.DataFrame(result)

print(result\_df)

1. **TÌM TRUNG VỊ CỦA MỖI CHỈ SỐ. TÌM TRUNG BÌNH VÀ ĐỘ LỆCH CHUẨN CỦA MỖI CHỈSỐ CHO CÁC CẦU THỦ TRONG TOÀN GIẢI VÀ CỦA MỖI ĐỘI. GHI KẾT QUẢ RA FILE RESULTS2.CSV, FORMAT NHƯ SAU:**



* **Code:**

import pandas as pd

*# Tải dữ liệu từ tệp CSV*

data = pd.read\_csv('results.csv')

*# Danh sách các chỉ số*

attributes = [

'matches\_played', 'starts', 'minutes', 'non\_Penalty\_Goals',

'Penalty Goals', 'Assists', 'Yellow\_Cards', 'Red\_Cards',

'xG\_x', 'npxG\_x', 'xAG\_x', 'PrgC', 'PrgP\_x', 'PrgR\_x',

'Gls\_x', 'Ast\_x', 'G+A', 'G-PK', 'G+A-PK', 'xG\_per\_90',

'xAG\_per\_90', 'xG+xAG', 'npxG\_per\_90', 'npxG+xAG\_per\_90',

'Gls\_y', 'Sh\_x', 'SoT', 'SoT%', 'Sh/90', 'SoT/90',

'G/Sh', 'G/SoT', 'Dist', 'FK', 'PK', 'PKatt\_y',

'xG\_y', 'npxG\_y', 'npxG/Sh', 'G-xG', 'np:G-xG',

'Total Cmp', 'Total Att', 'Total Cmp%', 'TotDist\_x',

'PrgDist', 'Short Cmp', 'Short Att', 'Short Cmp%',

'Medium Cmp', 'Medium Att', 'Medium Cmp%', 'Long Cmp',

'Long Att', 'Long Cmp%', 'Ast\_y', 'xAG\_y', 'xA',

'A-xAG', 'Key Passes', 'Final Third Passes', 'PPA',

'CrsPA', 'PrgP\_y', 'Pass Types.Live', 'Pass Types.Dead',

'Pass Types.FK', 'Pass Types.TB', 'Pass Types.Sw',

'Pass Types.Crs', 'Pass Types.TI', 'Pass Types.CK',

'Corner Kicks.In', 'Corner Kicks.Out', 'Corner Kicks.Str',

'Outcomes.Cmp', 'Outcomes.Off', 'Outcomes.Blocks',

'SCA', 'SCA90', 'PassLive', 'PassDead', 'TO',

'Sh\_y', 'Fld\_x', 'Def', 'GCA', 'GCA90',

'GCA PassLive', 'GCA PassDead', 'GCA TO', 'GCA Sh',

'GCA Fld', 'GCA Def', 'Touches', 'Def Pen',

'Def 3rd\_y', 'Mid 3rd\_y', 'Att 3rd\_y', 'Att Pen',

'Live', 'Att', 'Succ', 'Succ%', 'Tkld', 'Tkld% ',

'Carries', 'TotDist\_y', 'ProDist', 'ProgC',

'1/3', 'CPA', 'Mis', 'Dis', 'Rec', 'PrgR\_y',

'Starts', 'Mn/Start', 'Compl', 'Subs', 'Mn/Sub',

'unSub', 'PPM', 'onG', 'onGA', 'onxG',

'onxGA', 'Fls', 'Fld\_y', 'Off', 'Crs',

'OG', 'Recov', 'Aerial Won', 'Aerial Lost',

'Aerial Won%'

]

results = []

*# Tính toán cho toàn giải*

for attribute in attributes:

median\_value = data[attribute].median()

mean\_value = data[attribute].mean()

std\_value = data[attribute].std()

results.append({'Đội': 'all', 'Trung vị': median\_value, 'Giá trị trung bình ': mean\_value, 'Độ lệch chuẩn': std\_value})

*# Tính toán cho từng đội*

teams = data['Team'].unique()

for team in teams:

team\_data = data[data['Team'] == team]

for attribute in attributes:

median\_value = team\_data[attribute].median()

mean\_value = team\_data[attribute].mean()

std\_value = team\_data[attribute].std()

results.append({'Đội': team, 'Trung vị': median\_value, 'Giá trị trung bình': mean\_value, 'Độ lệch chuẩn': std\_value})

results\_df = pd.DataFrame(results)

results\_df.to\_csv('results2.csv', index=False)

1. **VẼ HISTORGRAM PHÂN BỐ CỦA MỖI CHỈSỐ CỦA CÁC CẦU THỦ TRONG TOÀN GIẢI VÀ MỖI ĐỘI.**

* **Code:**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import os

*# Đọc dữ liệu từ tệp CSV*

data = pd.read\_csv('results.csv')

*# Danh sách các chỉ số*

attributes = [

'matches\_played', 'starts', 'minutes', 'non\_Penalty\_Goals',

'Penalty Goals', 'Assists', 'Yellow\_Cards', 'Red\_Cards',

'xG\_x', 'npxG\_x', 'xAG\_x', 'PrgC', 'PrgP\_x', 'PrgR\_x',

'Gls\_x', 'Ast\_x', 'G+A', 'G-PK', 'G+A-PK', 'xG\_per\_90',

'xAG\_per\_90', 'xG+xAG', 'npxG\_per\_90', 'npxG+xAG\_per\_90',

'Gls\_y', 'Sh\_x', 'SoT', 'SoT%', 'Sh/90', 'SoT/90',

'G/Sh', 'G/SoT', 'Dist', 'FK', 'PK', 'PKatt\_y',

'xG\_y', 'npxG\_y', 'npxG/Sh', 'G-xG', 'np:G-xG',

'Total Cmp', 'Total Att', 'Total Cmp%', 'TotDist\_x',

'PrgDist', 'Short Cmp', 'Short Att', 'Short Cmp%',

'Medium Cmp', 'Medium Att', 'Medium Cmp%', 'Long Cmp',

'Long Att', 'Long Cmp%', 'Ast\_y', 'xAG\_y', 'xA',

'A-xAG', 'Key Passes', 'Final Third Passes', 'PPA',

'CrsPA', 'PrgP\_y', 'Pass Types.Live', 'Pass Types.Dead',

'Pass Types.FK', 'Pass Types.TB', 'Pass Types.Sw',

'Pass Types.Crs', 'Pass Types.TI', 'Pass Types.CK',

'Corner Kicks.In', 'Corner Kicks.Out', 'Corner Kicks.Str',

'Outcomes.Cmp', 'Outcomes.Off', 'Outcomes.Blocks',

'SCA', 'SCA90', 'PassLive', 'PassDead', 'TO',

'Sh\_y', 'Fld\_x', 'Def', 'GCA', 'GCA90',

'GCA PassLive', 'GCA PassDead', 'GCA TO', 'GCA Sh',

'GCA Fld', 'GCA Def', 'Touches', 'Def Pen',

'Def 3rd\_y', 'Mid 3rd\_y', 'Att 3rd\_y', 'Att Pen',

'Live', 'Att', 'Succ', 'Succ%', 'Tkld', 'Tkld% ',

'Carries', 'TotDist\_y', 'ProDist', 'ProgC',

'1/3', 'CPA', 'Mis', 'Dis', 'Rec', 'PrgR\_y',

'Starts', 'Mn/Start', 'Compl', 'Subs', 'Mn/Sub',

'unSub', 'PPM', 'onG', 'onGA', 'onxG',

'onxGA', 'Fls', 'Fld\_y', 'Off', 'Crs',

'OG', 'Recov', 'Aerial Won', 'Aerial Lost',

'Aerial Won%'

]

*#### Toàn giải*

output\_folder\_all = 'Toàn giải'

os.makedirs(output\_folder\_all, exist\_ok=True)

for attribute in attributes:

plt.figure(figsize=(10, 5))

sns.histplot(data[attribute], bins=40, kde=True, color='red')

plt.title(f'Bảng phân bố của chỉ số {attribute} trong toàn giải đấu')

plt.xlabel(attribute)

plt.ylabel('Tần suất')

plt.grid()

safe\_attribute = attribute.replace("/", "\_").replace(" ", "\_")

plt.savefig(f'{output\_folder\_all}/{safe\_attribute}\_league\_histogram.png')

plt.close()

*#### Từng đội một*

output\_folder\_teams = 'Từng đội'

os.makedirs(output\_folder\_teams, exist\_ok=True)

teams = data['Team'].unique()

for team in teams:

team\_folder = f'{output\_folder\_teams}/{team}'

os.makedirs(team\_folder, exist\_ok=True)

team\_data = data[data['Team'] == team]

for attribute in attributes:

plt.figure(figsize=(10, 5))

sns.histplot(team\_data[attribute], bins=40, kde=True, color='orange')

plt.title(f'Bảng phân bố chỉ số {attribute} của đội tuyển {team}')

plt.xlabel(attribute)

plt.ylabel('Tần suất')

plt.grid()

safe\_attribute = attribute.replace("/", "\_").replace(" ", "\_")

plt.savefig(f'{team\_folder}/{safe\_attribute}\_{team}\_histogram.png')

plt.close()

1. **TÌM ĐỘI BÓNG CÓ CHỈSỐ ĐIỂM SỐ CAO NHẤT Ở MỖI CHỈSỐ. THEO BẠN ĐỘI NÀO CÓ PHONG ĐỘ TỐT NHẤT GIẢI NGOẠI HẠNG ANH MÙA 2023-2024.**

* **Code:**

import pandas as pd

*# Đọc dữ liệu từ file CSV*

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

*# Danh sách các cột*

metrics = [

'matches\_played','starts','minutes','non\_Penalty\_Goals','Penalty Goals','Assists','Yellow\_Cards','Red\_Cards',

'xG\_x','npxG\_x','xAG\_x','PrgC','PrgP\_x','PrgR\_x','Gls\_x','Ast\_x','G+A','G-PK','G+A-PK','xG\_per\_90','xAG\_per\_90',

'xG+xAG','npxG\_per\_90','npxG+xAG\_per\_90','Gls\_y','Sh\_x','SoT','SoT%','Sh/90','SoT/90','G/Sh','G/SoT','Dist','FK','PK','PKatt\_y',

'xG\_y','npxG\_y','npxG/Sh','G-xG','np:G-xG','Total Cmp','Total Att','Total Cmp%','TotDist\_x','PrgDist','Short Cmp',

'Short Att','Short Cmp%','Medium Cmp','Medium Att','Medium Cmp%','Long Cmp','Long Att','Long Cmp%','Ast\_y','xAG\_y',

'xA','A-xAG','Key Passes','Final Third Passes','PPA','CrsPA','PrgP\_y','Pass Types.Live','Pass Types.Dead','Pass Types.FK','Pass Types.TB',

'Pass Types.Sw','Pass Types.Crs','Pass Types.TI','Pass Types.CK','Corner Kicks.In','Corner Kicks.Out','Corner Kicks.Str','Outcomes.Cmp',

'Outcomes.Off','Outcomes.Blocks','SCA','SCA90','PassLive','PassDead','TO','Sh\_y','Fld\_x','Def','GCA','GCA90','GCA PassLive','GCA PassDead','GCA TO','GCA Sh',

'GCA Fld','GCA Def','Touches','Def Pen','Def 3rd\_y','Mid 3rd\_y','Att 3rd\_y','Att Pen','Live','Att','Succ','Succ%','Tkld','Tkld% ' ,'Carries','TotDist\_y','ProDist','ProgC','1/3','CPA','Mis','Dis','Rec','PrgR\_y',

'Starts','Mn/Start','Compl','Subs','Mn/Sub','unSub','PPM','onG','onGA','onxG','onxGA','Fls','Fld\_y','Off','Crs','OG','Recov','Aerial Won','Aerial Lost','Aerial Won%'

]

*#### Tên đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số*

results = []

for metric in metrics:

if df[metric].notna().any():

top\_team\_index = df[metric].idxmax()

top\_team\_name = df.at[top\_team\_index, 'Team']

results.append({'Tên Đội': top\_team\_name, 'Chỉ Số': metric, 'Chỉ số điểm': df[metric][top\_team\_index]})

results\_df = pd.DataFrame(results)

print("Các đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số")

print(results\_df)

*#### Đội bóng có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024*

team\_counts = results\_df['Tên Đội'].value\_counts()

most\_team = team\_counts.idxmax()

print(f"Đội bóng có phong độ tốt nhất giải ngoại hạng Anh mùa 2023-2024 là: {most\_team}")

* Đội bóng có phong độ tốt nhất giải ngoại hạng Anh mùa 2023-2024 là: **MANCHESTER CITY**

**CÂU 3:**

1. **SỬ DỤNG THUẬT TOÁN K-MEANS ĐỂ PHÂN LOẠI CÁC CẦU THỦ THÀNH CÁC NHÓM CÓ CHỈ SỐ GIỐNG NHAU.**

* **Code:**

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

*# Đọc dữ liệu từ file CSV*

merged\_dfx = pd.read\_csv('results.csv')

*# Các chỉ số sử dụng để phân nhóm*

features = [

'matches\_played', 'starts', 'minutes', 'non\_Penalty\_Goals',

'Penalty Goals', 'Assists', 'Yellow\_Cards', 'Red\_Cards',

'xG\_x', 'npxG\_x', 'xAG\_x', 'PrgC', 'PrgP\_x', 'PrgR\_x',

'Gls\_x', 'Ast\_x', 'G+A', 'G-PK', 'G+A-PK', 'xG\_per\_90',

'xAG\_per\_90', 'xG+xAG', 'npxG\_per\_90', 'npxG+xAG\_per\_90',

'Gls\_y', 'Sh\_x', 'SoT', 'SoT%', 'Sh/90', 'SoT/90',

'G/Sh', 'G/SoT', 'Dist', 'FK', 'PK', 'PKatt\_y',

'xG\_y', 'npxG\_y', 'npxG/Sh', 'G-xG', 'np:G-xG',

'Total Cmp', 'Total Att', 'Total Cmp%', 'TotDist\_x',

'PrgDist', 'Short Cmp', 'Short Att', 'Short Cmp%',

'Medium Cmp', 'Medium Att', 'Medium Cmp%', 'Long Cmp',

'Long Att', 'Long Cmp%', 'Ast\_y', 'xAG\_y', 'xA',

'A-xAG', 'Key Passes', 'Final Third Passes', 'PPA',

'CrsPA', 'PrgP\_y', 'Pass Types.Live', 'Pass Types.Dead',

'Pass Types.FK', 'Pass Types.TB', 'Pass Types.Sw',

'Pass Types.Crs', 'Pass Types.TI', 'Pass Types.CK',

'Corner Kicks.In', 'Corner Kicks.Out', 'Corner Kicks.Str',

'Outcomes.Cmp', 'Outcomes.Off', 'Outcomes.Blocks',

'SCA', 'SCA90', 'PassLive', 'PassDead', 'TO',

'Sh\_y', 'Fld\_x', 'Def', 'GCA', 'GCA90',

'GCA PassLive', 'GCA PassDead', 'GCA TO', 'GCA Sh',

'GCA Fld', 'GCA Def', 'Touches', 'Def Pen',

'Def 3rd\_y', 'Mid 3rd\_y', 'Att 3rd\_y', 'Att Pen',

'Live', 'Att', 'Succ', 'Succ%', 'Tkld',

'Tkld% ', 'Carries', 'TotDist\_y', 'ProDist',

'ProgC', '1/3', 'CPA', 'Mis', 'Dis',

'Rec', 'PrgR\_y', 'Starts', 'Mn/Start', 'Compl',

'Subs', 'Mn/Sub', 'unSub', 'PPM', 'onG',

'onGA', 'onxG', 'onxGA', 'Fls', 'Fld\_y',

'Off', 'Crs', 'OG', 'Recov', 'Aerial Won',

'Aerial Lost', 'Aerial Won%'

]

*# Chọn các cột dữ liệu*

X = merged\_dfx[features]

*# Chuyển đổi các giá trị không hợp lệ thành NaN và sau đó loại bỏ*

X = X.apply(pd.to\_numeric, errors='coerce')

X = X.fillna(0)

*#Chuẩn hóa dữ liệu*

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X) #

*#Tìm số lượng cluster tối ưu*

inertia = []

for k in range(1, 11):

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

kmeans.fit(X\_scaled)

inertia.append(kmeans.inertia\_)

*# Vẽ đồ thị để xác định số lượng cluster*

plt.figure(figsize=(10, 6))

plt.plot(range(1, 11), inertia, marker='o')

plt.title('Phương pháp Elbow để xác định k tối ưu')

plt.xlabel('Số lượng cluster')

plt.ylabel('Inertia')

plt.grid(True)

plt.show()

*# Áp dụng K-means với số lượng cluster đã chọn*

optimal\_k = 3

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

merged\_dfx['Cluster'] = kmeans.fit\_predict(X\_scaled)

*# Gán tên cho từng nhóm*

cluster\_names = {

0: "Nhóm A - Cầu thủ tấn công",

1: "Nhóm B - Cầu thủ phòng ngự",

2: "Nhóm C - Cầu thủ Toàn Diện"

}

*# Gán tên nhóm vào cột mới trong DataFrame*

merged\_dfx['Cluster\_Name'] = merged\_dfx['Cluster'].map(cluster\_names)

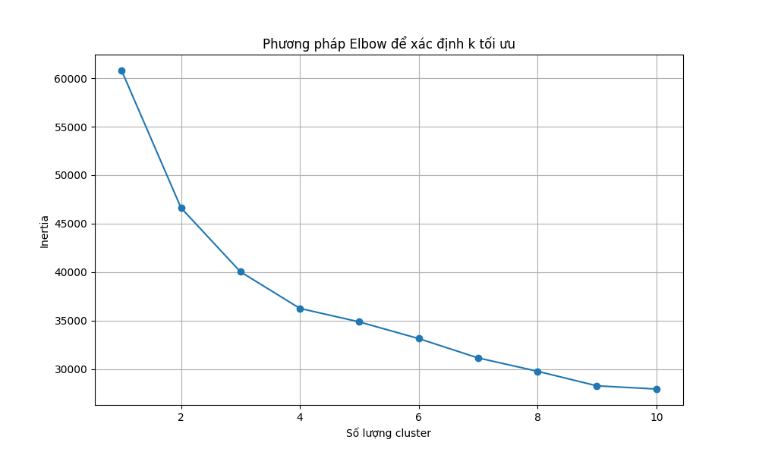
*# In ra tên cầu thủ theo từng nhóm*

for cluster in range(optimal\_k):

print(f"\n{cluster\_names[cluster]}:")

print(merged\_dfx[merged\_dfx['Cluster'] == cluster][['Name'] + features])

* Kết quả:

****

1. **THEO BẠN THÌ NÊN PHÂN LOẠI CẦU THỦ THÀNH BAO NHIÊU NHÓM? VÌ SAO? BẠN CÓ NHẬN XÉT GÌ VỀ KẾT QUẢ.**

* *Phân loại cầu thủ làm 3 nhóm:*
* **Nhóm Cầu thủ Tấn Công**: Vì một nhóm có thể gồm các cầu thủ có số bàn thắng cao, số pha kiến tạo nhiều, thường là tiền đạo.
* **Nhóm Cầu thủ Phòng Ngự**: Vì một nhóm khác có thể gồm các hậu vệ có số thẻ vàng hoặc số lần truy cản cao.
* **Nhóm Cầu thủ Toàn Diện**: Vì nhóm còn lại có thể là những cầu thủ có số trận đấu nhiều và chỉ số hỗ trợ tốt.
* *Nhận xét:*
* Quản lý sẽ dễ dàng, dễ hiểu.
* Ngắn gọn nhưng bao quát được tất cả thông tin cần thiết về mỗi cầu thủ.
* Tuy nhiên mức độ chính xác cũng hạn chế.

1. **SỬ DỤNG THUẬT TOÁN PCA, GIẢM SỐ CHIỀU DỮ LIỆU XUỐNG 2 CHIỀU, VẼ HÌNH PHÂN CỤM CÁC ĐIỂM DỮ LIỆU TRÊN MẶT 2D.**

* **Code:**

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

*# Đọc dữ liệu từ file CSV*

merged\_dfx = pd.read\_csv('results.csv')

*# Các chỉ số sử dụng để phân nhóm*

features = [

'matches\_played', 'starts', 'minutes', 'non\_Penalty\_Goals',

'Penalty Goals', 'Assists', 'Yellow\_Cards', 'Red\_Cards',

'xG\_x', 'npxG\_x', 'xAG\_x', 'PrgC', 'PrgP\_x', 'PrgR\_x',

'Gls\_x', 'Ast\_x', 'G+A', 'G-PK', 'G+A-PK', 'xG\_per\_90',

'xAG\_per\_90', 'xG+xAG', 'npxG\_per\_90', 'npxG+xAG\_per\_90',

'Gls\_y', 'Sh\_x', 'SoT', 'SoT%', 'Sh/90', 'SoT/90',

'G/Sh', 'G/SoT', 'Dist', 'FK', 'PK', 'PKatt\_y',

'xG\_y', 'npxG\_y', 'npxG/Sh', 'G-xG', 'np:G-xG',

'Total Cmp', 'Total Att', 'Total Cmp%', 'TotDist\_x',

'PrgDist', 'Short Cmp', 'Short Att', 'Short Cmp%',

'Medium Cmp', 'Medium Att', 'Medium Cmp%', 'Long Cmp',

'Long Att', 'Long Cmp%', 'Ast\_y', 'xAG\_y', 'xA',

'A-xAG', 'Key Passes', 'Final Third Passes', 'PPA',

'CrsPA', 'PrgP\_y', 'Pass Types.Live', 'Pass Types.Dead',

'Pass Types.FK', 'Pass Types.TB', 'Pass Types.Sw',

'Pass Types.Crs', 'Pass Types.TI', 'Pass Types.CK',

'Corner Kicks.In', 'Corner Kicks.Out', 'Corner Kicks.Str',

'Outcomes.Cmp', 'Outcomes.Off', 'Outcomes.Blocks',

'SCA', 'SCA90', 'PassLive', 'PassDead', 'TO',

'Sh\_y', 'Fld\_x', 'Def', 'GCA', 'GCA90',

'GCA PassLive', 'GCA PassDead', 'GCA TO', 'GCA Sh',

'GCA Fld', 'GCA Def', 'Touches', 'Def Pen',

'Def 3rd\_y', 'Mid 3rd\_y', 'Att 3rd\_y', 'Att Pen',

'Live', 'Att', 'Succ', 'Succ%', 'Tkld',

'Tkld% ', 'Carries', 'TotDist\_y', 'ProDist',

'ProgC', '1/3', 'CPA', 'Mis', 'Dis',

'Rec', 'PrgR\_y', 'Starts', 'Mn/Start', 'Compl',

'Subs', 'Mn/Sub', 'unSub', 'PPM', 'onG',

'onGA', 'onxG', 'onxGA', 'Fls', 'Fld\_y',

'Off', 'Crs', 'OG', 'Recov', 'Aerial Won',

'Aerial Lost', 'Aerial Won%'

]

*# Chọn các cột dữ liệu*

X = merged\_dfx[features]

*#Chuyển đổi các giá trị không hợp lệ thành NaN và sau đó loại bỏ*

X = X.apply(pd.to\_numeric, errors='coerce')

X = X.fillna(0)

*#Chuẩn hóa dữ liệu*

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

*# Áp dụng K-means với số lượng cluster đã chọn*

optimal\_k = 3

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

merged\_dfx['Cluster'] = kmeans.fit\_predict(X\_scaled)

*# Gán tên cho từng nhóm*

cluster\_names = {

0: "Nhóm A - Cầu thủ tấn công",

1: "Nhóm B - Cầu thủ phòng ngự",

2: "Nhóm C - Cầu thủ Toàn Diện"

}

*#Gán tên nhóm vào cột mới trong DataFrame*

merged\_dfx['Cluster\_Name'] = merged\_dfx['Cluster'].map(cluster\_names)

*# Thực hiện PCA để giảm số chiều về 2*

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

*# Vẽ đồ thị phân cụm trên mặt phẳng 2D*

plt.figure(figsize=(10, 5))

for cluster in range(optimal\_k):

plt.scatter(X\_pca[merged\_dfx['Cluster'] == cluster, 0],

X\_pca[merged\_dfx['Cluster'] == cluster, 1],

label=cluster\_names[cluster])

plt.title('Phân cụm cầu thủ trên mặt phẳng 2D')

plt.xlabel('PC 1')

plt.ylabel('PC 2')

plt.legend()

plt.grid(True)

plt.show()

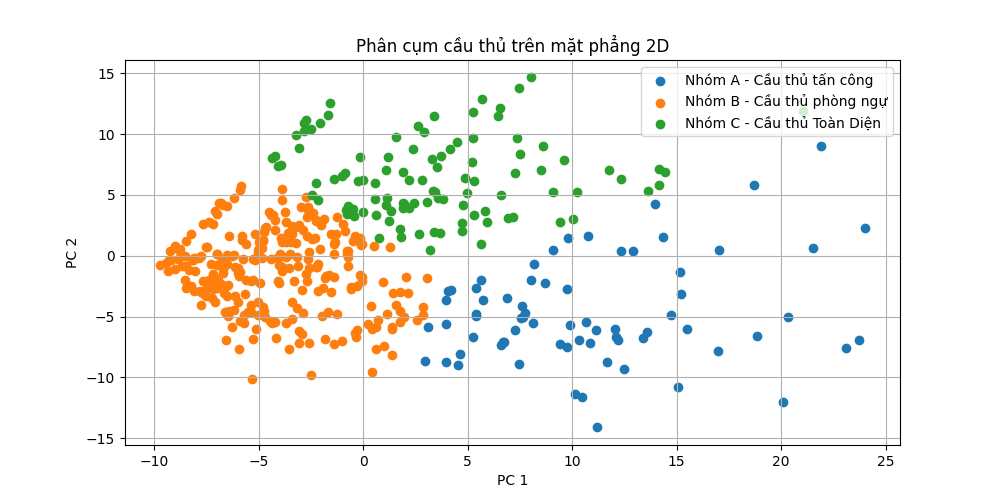
*# In ra tên cầu thủ theo từng nhóm*

for cluster in range(optimal\_k):

print(f"\n{cluster\_names[cluster]}:")

print(merged\_dfx[merged\_dfx['Cluster'] == cluster][['Name'] + features])

* Kết quả:

****

1. **VIẾT CHƯƠNG TRÌNH PYTHON VẼ BIỂU ĐỒ RADA (RADAR CHART) SO SÁNH CẦU THỦ VỚI ĐẦU VÀO NHƯ SAU:**

* **Code:**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import os

*# Đọc dữ liệu từ file CSV*

data = pd.read\_csv('results.csv')

*# Chọn các chỉ số cần so sánh*

attributes = [

'matches\_played', 'starts', 'minutes', 'non\_Penalty\_Goals', 'Penalty Goals', 'Assists',

'Yellow\_Cards', 'Red\_Cards', 'xG\_x', 'npxG\_x', 'xAG\_x', 'PrgC', 'PrgP\_x', 'PrgR\_x',

'Gls\_x', 'Ast\_x', 'G+A', 'G-PK', 'G+A-PK', 'xG\_per\_90', 'xAG\_per\_90', 'xG+xAG',

'npxG\_per\_90', 'npxG+xAG\_per\_90', 'Gls\_y', 'Sh\_x', 'SoT', 'SoT%', 'Sh/90', 'SoT/90',

'G/Sh', 'G/SoT', 'Dist', 'FK', 'PK', 'PKatt\_y', 'xG\_y', 'npxG\_y', 'npxG/Sh',

'G-xG', 'np:G-xG', 'Total Cmp', 'Total Att', 'Total Cmp%', 'TotDist\_x', 'PrgDist',

'Short Cmp', 'Short Att', 'Short Cmp%', 'Medium Cmp', 'Medium Att', 'Medium Cmp%',

'Long Cmp', 'Long Att', 'Long Cmp%', 'Ast\_y', 'xAG\_y', 'xA', 'A-xAG', 'Key Passes',

'Final Third Passes', 'PPA', 'CrsPA', 'PrgP\_y', 'Pass Types.Live', 'Pass Types.Dead',

'Pass Types.FK', 'Pass Types.TB', 'Pass Types.Sw', 'Pass Types.Crs', 'Pass Types.TI',

'Pass Types.CK', 'Corner Kicks.In', 'Corner Kicks.Out', 'Corner Kicks.Str',

'Outcomes.Cmp', 'Outcomes.Off', 'Outcomes.Blocks', 'SCA', 'SCA90', 'PassLive',

'PassDead', 'TO', 'Sh\_y', 'Fld\_x', 'Def', 'GCA', 'GCA90', 'GCA PassLive',

'GCA PassDead', 'GCA TO', 'GCA Sh', 'GCA Fld', 'GCA Def', 'Touches', 'Def Pen',

'Def 3rd\_y', 'Mid 3rd\_y', 'Att 3rd\_y', 'Att Pen', 'Live', 'Att', 'Succ', 'Succ%',

'Tkld', 'Tkld% ', 'Carries', 'TotDist\_y', 'ProDist', 'ProgC', '1/3', 'CPA', 'Mis',

'Dis', 'Rec', 'PrgR\_y', 'Starts', 'Mn/Start', 'Compl', 'Subs', 'Mn/Sub', 'unSub',

'PPM', 'onG', 'onGA', 'onxG', 'onxGA', 'Fls', 'Fld\_y', 'Off', 'Crs', 'OG',

'Recov', 'Aerial Won', 'Aerial Lost', 'Aerial Won%'

]

*# Kiểm tra xem các thuộc tính có trong dữ liệu không*

attributes = [attr for attr in attributes if attr in data.columns]

*# Tạo thư mục để lưu hình ảnh*

luu = 'radarChartPlot'

os.makedirs(luu, exist\_ok=True)

*# So sánh tất cả các cầu thủ*

for i in range(len(data)):

for j in range(i + 1, len(data)):

player1 = data.iloc[i]

player2 = data.iloc[j]

*# Số lượng các chỉ số*

num\_vars = len(attributes)

*# Thiết lập góc cho mỗi chỉ số*

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

*# Đưa điểm đầu tiên về để khép kín hình*

values1 = player1[attributes].values.flatten().tolist()

values1 += values1[:1]

values2 = player2[attributes].values.flatten().tolist()

values2 += values2[:1]

angles += angles[:1]

*# Tạo hình vẽ radar*

fig, ax = plt.subplots(figsize=(12, 6), subplot\_kw=dict(polar=True))

ax.fill(angles, values1, color='red', alpha=0.3)

ax.fill(angles, values2, color='blue', alpha=0.3)

*# Đường viền cho cầu thủ*

ax.plot(angles, values1, color='red', linewidth=1, label=player1["Name"])

ax.plot(angles, values2, color='blue', linewidth=1, label=player2["Name"])

*# Trục và chỉ số*

ax.set\_yticklabels([], fontsize=5)

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

ax.set\_xticklabels(attributes, fontsize=5)

*# Tiêu đề*

plt.title(f'{player1["Name"]} và {player2["Name"]}', size=20, color='black', weight='bold')

plt.legend(loc='upper right', bbox\_to\_anchor=(0.1, 0.1))

*# Lưu biểu đồ vào thư mục với đường dẫn đầy đủ*

file\_name = f'{player1["Name"].replace("/", "\_")}\_vs\_{player2["Name"].replace("/", "\_")}.png'

plt.savefig(os.path.join(luu, file\_name))

plt.close()

*# In ra kết quả*

print("Tất cả hình ảnh đã được lưu vào thư mục:", luu)