

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO BÀI TẬP LỚN

LẬP TRÌNH VỚI PYTHON

Giảng viên hướng dẫn: GV.Kim Ngọc Bách

Họ và tên	Tạ Quang Chiến
Mã sinh viên	B22DCCN109
Lớp	D22CQCN01-B
Nhóm	11

Mục lục

I. Cơ sở	2
1. Yêu cầu và cách chạy báo cáo	2
a) Yêu cầu	2
b) Cách chạy báo cáo.....	2
2. Xây dựng cơ sở cho bài toán.....	4
a) Cấu trúc thư mục	4
b) Xây dựng file object.....	4
II. Tiến hành thực hiện	7
1. Bài tập 1	7
a) Chuẩn bị các thư viện cần thiết	7
b) Vấn đề thu thập dữ liệu từ website.....	7
c) Thu thập dữ liệu từ từng trang web.....	10
d) Lọc,lưu kết quả và hoàn thành	12
2. Bài tập 2	15
a) Chuẩn bị dữ liệu	15
b) Tìm top 3 cầu thủ có chỉ số cao nhất và thấp nhất ở mọi chỉ số.....	16
c) 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	18
d) Vẽ histogram	21
e) Tìm đội bóng có chỉ số điểm số cao nhất ở mọi chỉ số.....	23
3. Bài tập 3	26
a) Các thư viện cần thiết.....	26
b) Lọc và chuẩn hóa dữ liệu	27
c) Tìm số điểm hội tụ cho thuật toán Kmean.....	29
d) Phân cụm Kmean.....	31
e) Sử dụng phương pháp giảm chiều của dữ liệu bằng thuật toán PCA	32
g) So sánh 2 cầu thủ với radarChart.....	33
4. Bài tập 4	38
a) Update file object	38
b) Thu thập thông tin cầu thủ	39
c) Đề xuất phương án định giá cầu thủ	43

I. Cơ sở

1. Yêu cầu và cách chạy báo cáo

a) Yêu cầu

- 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'
- 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. Đánh giá xem đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024
- Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau.
- Nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? 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
- 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.

b) Cách chạy báo cáo

Bước 1: Clone thư mục

Bước 2: Chạy các file bt1,bt2,bt3,bt4

Một số điểm cần lưu ý:

*File bt1 sinh ra các file result.csv theo yêu cầu đề bài, ngoài ra nó còn sinh ra 2 file là players.pkl và squads.pkl để phục vụ các bài sau đó.

*Đối với bài2 và bài 3 các yêu cầu được thực hiện qua từng hàm ở cuối file

*File bt2 sinh ra các file result2 theo yêu cầu đề bài(tìm mean-median-stddev)ngoài ra nó còn sinh ra file result3(Tìm top 3 cầu thủ cho mỗi thuộc tính),phần tìm đội tốt nhất thực hiện trên cửa sổ terminal.

*Phần vẽ histogram chương trình chạy tương đối lâu nên đã được vẽ trước và lưu trong thư mục hình vẽ các, hàm vẽ sẽ được comment lại

```
print("Draw histogram -----")  
# drawHistogram()
```

Có thể bỏ comment nếu muốn thực hiện vẽ lại

*File 3 thực hiện các yêu cầu trên cửa sổ terminal

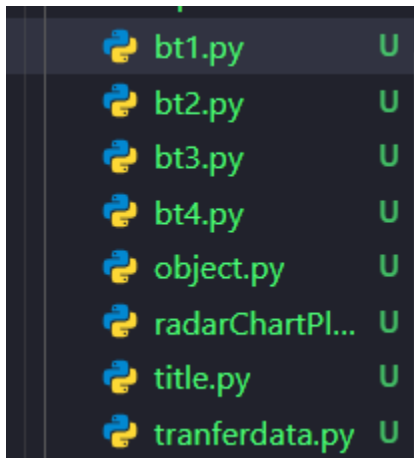
*File bt4 lấy dữ liệu từ 2 file tranfers.pkl và slugAndAge.pkl, 2 file này do chạy file tranferdata.py, 2 file dữ liệu này đã nằm trong thư mục code nên không cần phải chạy lại file tranferdata.py nữa. Có thể thực hiện chạy lại nhưng chạy khá lâu(quét khoảng 400 link cầu thủ) và file yêu cầu chạy đa luồng

*Các file còn lại chỉ chứa hàm, mảng và lớp dùng chung

****Nếu loại bỏ hết các file dữ liệu thì sau đây là thứ tự chạy
bt1.py =>bt2.py=>bt3.py=>tranferdata.py=>bt4**

2. Xây dựng cơ sở cho bài toán

a) Cấu trúc thư mục



Phần báo cáo được chia thành 8 file chính bao gồm:

- + file bt1,bt2,bt3,bt4 dùng để đáp ứng các yêu cầu của phần yêu cầu đề bài
- + file object dùng để lưu các đối tượng có trong bài(gồm có đối tượng Player(cầu thủ) và Squad(cho đội bóng))
- +file title chứa mảng, hàm để lấy tiêu đề của thuộc tính, thuộc tính của các đối tượng sẽ giải thích chi tiết ở phần sau
- + file radarCharPlot chứa hàm dùng để thực hiện vẽ sơ đồ phục vụ cho yêu cầu bài 3
- + file tranferdata dùng để lấy dữ liệu từ trang web tranferfootball phục vụ cho yêu cầu bài 4

b) Xây dựng file object

File này chứa các lớp cơ bản và là cơ sở để xây dựng các bài toán sau này

Chứa các thành phần:

- Class Player(cầu thủ): chứa đầy đủ các thuộc tính mà đề bài yêu cầu
 - o Ngoài ra còn 1 số hàm set để thuận tiện cho việc thiết lập chỉ số cầu thủ
 - o Hàm toString để thuận tiện cho việc debug và đọc ghi file

```

class Player:
    def __init__(self, name, nation, position, team, age):
        self.name = name
        self.nation = nation if nation != "" else "N/a"
        self.team = team
        self.position = position
        self.age = int(age) if age != "N/a" else 0

        self.playing_time = {
            "matches_played": "N/a",
            "starts": "N/a",
            "minutes": 0
        }

    # Performance
    self.performance = {
        "non_penalty_goals": "N/a",
        "penalty_goals": "N/a",
        "assists": "N/a",
        "yellow_cards": "N/a",
        "red_cards": "N/a"
    }

```

Hình 1: một số thuộc tính của class Player

- Class Squad: chứa các thông tin tương tự với Player
- Các lớp quản lý cho 2 class Player và Squad lần lượt là Player_Manager và Squad_Manager
 - o Cả 2 lớp đều có phương thức add để thêm cầu thủ/đội bóng vào trong danh sách quản lý của chúng
 - o Các Squad phân biệt với nhau bởi tên, trong khi đó Player thì được phân biệt bởi Tên và đội, do trong kì chuyển nhượng của mùa giải các cầu thủ được chuyển qua đội tuyển khác(trong trường hợp này coi như là 2 người khác nhau để dễ dàng đánh giá, các cầu thủ đã được kiểm tra để đảm bảo không có 2 cầu thủ trùng tên trong 1 đội)
 - o Lớp Player còn có 1 hàm filtering để lọc cầu thủ chơi trên 90 và sortByName để sắp xếp cầu thủ theo tên

```

class Player_Manager:
    def __init__(self) -> None:
        self.list_player=[]

    def add_Player(self,player):
        self.list_player.append(player)

    def findPlayerByNameandTeam(self,name,team):
        for i in self.list_player:
            if i.name==name and i.team==team:
                return i
        return None

    def filtering(self):
        self.list_player=list(filter(lambda p:p.playing_time["minutes"]>90,self.list_player))

    def sortingByName(self):
        self.list_player=sorted(self.list_player,key=lambda x:(x.name,-x.age))

class Squad_Manager:
    def __init__(self) -> None:
        self.list_squad=[]

    def add_Squad(self,squad):
        self.list_squad.append(squad)

    def findSquadByName(self,name):
        for i in self.list_squad:
            if i.name==name:
                return i
        return None

```

II. Tiến hành thực hiện

1. Bài tập 1

a) Chuẩn bị các thư viện cần thiết

```
1 from selenium import webdriver
2 from selenium.webdriver.chrome.service import Service
3 from webdriver_manager.chrome import ChromeDriverManager
4 import time
5 from bs4 import BeautifulSoup
6 driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
7
8 # Truy cập vào trang web
9 from object import Player
10 from object import Player_Manager
11 from object import Squad
12 from object import Squad_Manager
13
14 player_manager=Player_Manager()
15 squad_manager=Squad_Manager()
16
```

Cần thư viện selenium để lấy file html từ trang web

Thư viện BeautifulSoup để thực hiện phân tích file html thành các thành phần

Import các class từ file object để nhập thông tin

```
def validdata(n):
    if n=='': return "N/a"
    return float(n)
```

Hàm này thực hiện chuẩn hóa các dữ liệu không hợp lệ

b) Vấn đề thu thập dữ liệu từ website

- Có tổng cộng 10 trang web để có thể lấy đầy đủ dữ liệu cho 1 đối tượng Player/Squad

- Mỗi trang web sử dụng 1 url riêng, tên các bảng trên web riêng, số lượng thuộc tính của từng đối tượng cũng như tên phân biệt cho các dạng dữ liệu đó. Xong tất cả các trang web đều có các thuộc tính như vậy, ta xây dựng 1 hàm để xử lý việc lấy dữ liệu từ từng url:

```
def getDataFromWeb(url, idPlayerTable, idSquadTable, lengthPlayerData, DataName):
```


Với các tham số

+ url: Địa chỉ trang web

+idPlayerTable: id của bảng cầu thủ trong thẻ html của trang web

+idSquadTable: : id của bảng đội bóng trong thẻ html của trang web(dữ liệu nay không cần thiết ở bài tập này nhưng ta vẫn thu thập thông tin để tránh lặp lại việc lấy dữ liệu cho các bài sau)

+lengthPlayerData: số lượng thuộc tính của đối tượng Player/squad trong bảng

+DataName: tên gọi chung của các thuộc tính có trong trang web

* Hàm trả về 2 mảng dữ liệu 1 cho đối tượng Player, 1 cho đối tượng class

```
resultPlayerData=[]  
resultSquadData=[]
```

Bắt đầu với hàm `driver.get(url)`, để truy cập vào trang web,ta chờ khoảng 3s để tất cả các nội dung trên trang được tải hết,ta chia file html thu được ra thành 2 thành phần của Player và Squad

- Bắt đầu với bảng Player

```
driver.get(url)  
try:  
    time.sleep(3)  
  
    # Lấy toàn bộ nội dung HTML sau khi trang đã tải xong  
    html_content = driver.page_source  
  
    soup = BeautifulSoup(html_content, 'html.parser')  
  
    #player  
    div_stats = soup.find('div', id=idPlayerTable)  
    table = div_stats.find('table')  
    tbody = table.find('tbody')  
    tr_list=tbody.find_all('tr')
```





- Qua các bước sơ chế, các phương thức của thư viện BeautifulSoup ta thu được thành phẩm là 1 list, mỗi phần tử trong list đó chính là dãy các thuộc tính của 1 đối tượng Player thu được từ url đó, t bắt đầu lặp hết để xử lý các đoạn dữ liệu

```
miss=25
for ind,i in enumerate(tr_list):
    if(ind==miss):
        miss=miss+25+1
        continue
    arr=[]
    for index, value in enumerate(i.find_all('td')):
        if(index==1):
            a=value.find('a')
            if(a is not None):
                span1=a.find('span', recursive=False)
                span2=span1.find("span")
                span2.extract()
            s=value.text.strip()

            if index>=4 and index!=lengthPlayerData:
                s=s.replace(",","")
                s=validdata(s)
            arr.append(s)
    arr.pop() # Loại bỏ giá trị không liên quan
    resultPlayerData.append(arr)
```

Qua xử lý, chuẩn hóa dữ liệu(loại bỏ dấu phẩy của các chỉ số) ta thu được dãy là các thuộc tính của đối tượng Player, 1 dãy các thuộc tính lại được thêm vào mảng resultPlayerData, có nghĩa là mảng resultPlayerData chứa n phần tử tương ứng với n đối tượng Player, mỗi phần tử là thuộc tính của đối tượng Player

******(Biến miss dùng để đánh dấu mỗi 25 đối tượng player được xử lý thì sẽ bỏ qua 1 dòng do cấu trúc của bảng trên trang web cứ mỗi 25 cầu thủ lại lặp lại hàng header thuộc tính)

43	Eusebio Oliveira		MF	Wesley Marshall	23	1997	31	2,677	77	69.3	26.7	60	64	10	5	43	1	1.40	76	42	-73	-0.13	-0.32	33.9	42.0	-13.8	-0.32	-0.07
24	Julián Álvarez		MF	Manchester City	23	2000	36	2,647	74	77.4	29.4	31	83	15	5	13	2	2.42	76	24	+52	+1.77	+0.60	61.4	25.9	+35.5	+1.21	+0.11
25	Steven Alzate		MF	Brighton	24	1998	0					0		0	0		1											
Rk	Player	Nation	Pos	Squad	Age	Born	MP	Min	Mn/MP	Min%	90s	Starts	Mn/Start	Compl	Subs	Mn/Sub	unSub	PPM	onG	onGA	+/-	+/-90	On-Off	onxG	onxGA	xG+/-	xG+/-90	On-Off
26	Harry Amass		DF	Manchester Utd	16	2007	0					0		0	0		6											
27	Zeki Amdouni		FW	Burnley	22	2000	34	1,953	57	57.1	21.7	27	69	2	7	14	4	0.62	20	41	-21	-0.97	+0.01	19.5	36.2	-16.7	-0.77	+0.03

*** (biến lengthPlayerData dùng để đánh dấu thuộc tính cuối cùng của bảng Player do phần cuối của thẻ tr là thẻ link, không liên quan đến thuộc tính của đối tượng)*

- Tương tự Player, phần lấy dữ liệu cho Squad cũng có cấu trúc tương tự như vậy

```
#squad
table = soup.find('table', id=idSquadTable)
tbody = table.find('tbody')
tr_list=tbody.find_all('tr')

miss=25
for ind,i in enumerate(tr_list):
    arr=[]
    th=i.find("th")
    arr.append(th.text.strip())
    for index, value in enumerate(i.find_all('td')):
        s=value.text.strip()
        if index!=0:
            s=s.replace(",","")
            s=validdata(s)
        arr.append(s)
    resultSquadData.append(arr)
except :
    print("Something went wrong")
finally:
    print("Finish Page "+DataName)
return [resultPlayerData,resultSquadData]
```

Cuối cùng ta in ra dòng chữ finish để đánh dấu đã hoàn thành thu thập dữ liệu tại trang web, đồng thời trả lại 2 mảng kết quả resultPlayerData và resultSquadData

c) Thu thập dữ liệu từ từng trang web

- Khởi đầu với trang web chứa đầy đủ tất cả các cầu thủ nhất là Playing Time, chứa đầy đủ hơn 760 cầu thủ làm gốc cho bài toán

-Bởi vì trong phần quản lý mảng của 2 đối tượng quản lý đang rỗng nên mỗi cầu thủ hoặc đội bóng sẽ gọi hàm tạo mới và được lưu vào trong mảng để các lớp quản lý lưu trữ

```

#start get Player and Team*****

url = 'https://fbref.com/en/comps/9/2023-2024/playingtime/2023-2024-Premier-League-Stats'
idPlayerTable="div_stats_playing_time"
lengthPlayerData=28
idSquadTable="stats_squads_playing_time_for"
lengthSquadData=35
DataName="Playing Time"
list_player_result,list_squad_result=getDataFromWeb(url,idPlayerTable,idSquadTable,lengthPlayerData,DataName)

#squad data
for i in list_squad_result:
    s=squad_manager.findSquadByName(i[0])
    if s==None:
        new_s=Squad(*i[0:3])
        new_s.setPlayingTimeDetail(i[8:11],i[11:14],i[14:17],i[19:21])
        squad_manager.add_Squad(new_s)

#Player data
for i in list_player_result:
    p=player_manager.findPlayerByNameandTeam(i[0],i[3])
    s=squad_manager.findSquadByName(i[3])
    if p==None:
        new_p=Player(i[0],i[1],i[2],i[3],i[4])
        new_p.setPlayingTimeDetail(i[11:14],i[14:17],i[17:20],i[23:25])
        player_manager.add_Player(new_p)
        s.players.append(new_p)

#End get Player and Team*****

```

Các đối tượng được thiết lập các chỉ số sau đó được lưu vào lớp quản lý mảng đối tượng tương ứng,

Ngoài ra, cầu thủ còn được thêm vào danh sách các cầu thủ có trong một đội bóng là cơ sở để thực hiện các bài toán sau này

Các url được thực hiện lần lượt cho đến hết 10 url mà bài toán yêu cầu

```

url="https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats"
idPlayerTable="div_stats_standard"
lengthPlayerData=35
idSquadTable="stats_squads_standard_for"
lengthSquadData=32
DataName="Standard"
list_player_result,list_squad_result=getDataFromWeb(url,idPlayerTable,idSquadTable,lengthPlayerData,DataName)

#squad data
for i in list_squad_result:
    s=squad_manager.findSquadByName(i[0])
    if s!=None:
        s.poss=i[4]
        s.setPlaying_time(i[4:7])
        s.setPerformance([i[11],i[12],i[9],i[14],i[15]])
        s.setExpected(i[16:19])
        s.setProgression(i[20:22])
        s.setPer90(i[22:])

#player data
for i in list_player_result:
    p=player_manager.findPlayerByNameandTeam(i[0],i[3])
    if p!=None:
        p.setPlaying_time(i[6:9])
        p.setPerformance([i[13],i[14],i[11],i[16],i[17]])
        p.setExpected(i[18:21])
        p.setProgression(i[22:25])
        p.setPer90(i[25:])

```

Ví dụ thu thập dữ liệu cho url dẫn đến trang web standard

Mỗi url, ta thu được từng loại dữ liệu, mỗi loại này được thiết lập vào trong thuộc tính của đối tượng Player thông qua các hàm set của lớp, các thuộc tính được set theo mảng dữ liệu tương ứng với nó để tích kiệm thời gian(ví dụ như hàm setPlaying_time,setPerformance,... ở hình trên. Tương tự với các url còn lại

d) Lọc,lưu kết quả và hoàn thành

- Sau khi thu thập tất cả các đối tượng Player/Squad tiến hành lưu kết quả vào trong file players.pkl và squads.pkl để phục vụ cho các phần phía sau

```

import pickle
file_path_players=os.path.join(os.path.dirname(__file__), "players.pkl")
file_path_squads=os.path.join(os.path.dirname(__file__), "squads.pkl")
file_path_result=os.path.join(os.path.dirname(__file__), "result.csv")

# Ghi các đối tượng vào file
with open(file_path_players, "wb") as file:
    pickle.dump(player_manager.list_player, file)

with open(file_path_squads, "wb") as file:
    pickle.dump(squad_manager.list_squad, file)
print("Ghi thành công tất cả player vào file players.pkl và squads.pkl!")

```

- Tiến hành lọc cùng với đó là sắp xếp lại các Player có thời lượng chơi trên 90 trong mùa giải bằng phương thức filtering và sortByName của lớp quản lý đối tượng Player

```
player_manager.filtering()
player_manager.sortingByName()
```

- Các đối tượng được lưu vào file csv thông qua thư viện csv

```
import csv
from title import header
from title import row
with open(file_path_result, mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(header)

    for player in player_manager.list_player:
        r=row(player)
        writer.writerow(r)
print("Ghi thành công vào file result.csv", "Exam 1 Success", sep="\n")
import subprocess
# Mở file CSV
# subprocess.Popen(["start", file_path_result], shell=True)
os.startfile(file_path_result)
```

- Mảng header được import từ file title với từng phần tử là tiêu đề của thuộc tính của các đối tượng Player

```
header = [
    "name", "nation", "team", "position", "age",
    # Playing Time
    "matches_played", "starts", "minutes",
    # Performance
    "non_penalty_goals", "penalty_goals", "assists", "yellow_cards", "red_cards",
    # Expected
    "xG", "npxG", "xAG",
    # Progression
    "PrgC", "PrgP", "PrgR",
    # Per 90 minutes
    "per90_Gls", "per90_Ast", "per90_G+A", "per90_G-PK", "per90_G+A-PK",
    "per90_xG", "per90_xAG", "per90_xG+xAG", "per90_npxG", "per90_npxG+xAG",
    # Goalkeeping
    "GA", "GA90", "SoTA", "Saves", "Save%", "W", "D", "L", "CS", "CS%",
    # Goalkeeping Penalty Kicks
    "PKatt", "PKA", "PKsv", "PKm", "GK_Save%",
    # Shooting Standard
    "Gls", "Sh", "SoT", "SoT%", "Sh/90", "SoT/90", "G/Sh", "G/SoT", "Dist", "FK", "PK", "PKatt",
    # Shooting Expected
    "xG", "npxG", "npxG/Sh", "G-xG", "np:G-xG",
    # Passing Total
    "Pass_Cmp", "Pass_Att", "Pass_Cmp%", "TotDist", "PrgDist",
    # Passing Short
    "Short_Cmp", "Short_Att", "Short_Cmp%",
    # Passing Medium
    "Medium_Cmp", "Medium_Att", "Medium_Cmp%",
    # Passing Long
```

- Sau cùng ta mở file result.csv hoàn thành yêu cầu bài 1 của bài toán

[illegible]

2. Bài tập 2

a) Chuẩn bị dữ liệu

```
import pickle
import csv
import statistics
from title import header,row,row2,header2,rowsquad
import os
file_path_players = os.path.join(os.path.dirname(__file__), "players.pkl")
file_path_squads = os.path.join(os.path.dirname(__file__), "squads.pkl")
file_path_resultTop3 = os.path.join(os.path.dirname(__file__), "result3.csv")
file_path_result2 = os.path.join(os.path.dirname(__file__), "result2.csv")
# Đọc các đối tượng từ file
list_player=[]
with open(file_path_players, "rb") as file:
    list_player = pickle.load(file)

list_squad=[]
with open(file_path_squads, "rb") as file:
    list_squad = pickle.load(file)
print("Load data success!")

NUMBER_OF_ATTR=167 #loại bỏ 5 thuộc tính đầu
```

Ta lấy 2 mảng đối tượng Player và Squad từ kết quả của bài 1

NUMBER_OF_ATTR là số thuộc tính chính của đối tượng Player/Squad, thực tế là có 172 thuộc tính nhưng ta chỉ lấy 167 thuộc tính(loại bỏ 5 thuộc tính đầu) có thể so sánh được

* Trong file title, ta xây dựng hàm row, hàm row nhận tham số là 1 đối tượng Player, trả về 1 mảng là giá trị các thuộc tính của đối tượng Player


```

def row(player):
    return [
        player.name, player.nation, player.team, player.position, player.age,
        # Playing Time
        player.playing_time["matches_played"], player.playing_time["starts"], player.playing_time["minutes"],
        # Performance
        player.performance["non_penalty_goals"], player.performance["penalty_goals"], player.performance["assists"],
        player.performance["yellow_cards"], player.performance["red_cards"],
        # Expected
        player.expected["xG"], player.expected["npxG"], player.expected["xAG"],
        # Progression
        player.progression["PrgC"], player.progression["PrgP"], player.progression["PrgR"],
        # Per 90 minutes
        player.per_90["Gls"], player.per_90["Ast"], player.per_90["G+A"], player.per_90["G-PK"],
        player.per_90["G+A-PK"], player.per_90["xG"], player.per_90["xAG"], player.per_90["xG+xAG"],
        player.per_90["npxG"], player.per_90["npxG+xAG"],
        # Goalkeeping Performance
        player.goalkeeping["Performance"]["GA"], player.goalkeeping["Performance"]["GA90"],
        player.goalkeeping["Performance"]["SoTA"], player.goalkeeping["Performance"]["Saves"],
        player.goalkeeping["Performance"]["Save%"], player.goalkeeping["Performance"]["W"],
        player.goalkeeping["Performance"]["D"], player.goalkeeping["Performance"]["L"],
        player.goalkeeping["Performance"]["CS"], player.goalkeeping["Performance"]["CS%"],
        # Goalkeeping Penalty Kicks
        player.goalkeeping["Penalty Kicks"]["PKatt"], player.goalkeeping["Penalty Kicks"]["PKA"],
        player.goalkeeping["Penalty Kicks"]["PKsv"], player.goalkeeping["Penalty Kicks"]["PKm"],
        player.goalkeeping["Penalty Kicks"]["Save%"],
        # Shooting Standard
        player.shooting["Standard"]["Gls"], player.shooting["Standard"]["Sh"],
        player.shooting["Standard"]["SoT"], player.shooting["Standard"]["SoT%"],
        player.shooting["Standard"]["Sh/90"], player.shooting["Standard"]["SoT/90"],
        player.shooting["Standard"]["G/Sh"], player.shooting["Standard"]["G/SoT"],
        player.shooting["Standard"]["Dist"], player.shooting["Standard"]["FK"],
        player.shooting["Standard"]["PK"], player.shooting["Standard"]["PKatt"],
    ]

```

b) Tìm top 3 cầu thủ có chỉ số cao nhất và thấp nhất ở mọi chỉ số

- Từ list_player ta lấy hết tất cả các giá trị thuộc tính của các đối tượng Player sau đó lưu vào mảng to_compare_list, các giá trị không hợp lệ của cầu thủ sẽ được chuyển đổi thành 0 để dễ dàng so sánh

```

from common import row
to_compare_list=[]
for i in list_player:
    all_attr_of_player=row(i)
    for index,value in enumerate(all_attr_of_player):
        if(index<5): continue
        if value=="N/a":
            all_attr_of_player[index]=0
    to_compare_list.append(all_attr_of_player)

```

-Vì số lượng thuộc tính tương đối lớn, việc xem kết quả qua cửa sổ terminal tương đối bất tiện nên thông tin về top chỉ số cao, thấp nhất sẽ được đưa vào file csv để dễ dàng quan sát

-Sử dụng hàm sort cho từng mảng trong to_compare_list, với tiêu chí sắp xếp là giá trị của thuộc tính thứ 5+i hay phần tử thứ 5+i trong từng phần tử của mảng to_compare_list

-Mảng header lại tiếp tục được sử dụng để làm đầu đề cho các top 3 về thuộc tính đó

```
with open(file_path_resultTop3, mode='w', newline='', encoding='utf-8') as file:#file result3 lưu top 3 các cầu thủ có thuộc tính
    writer = csv.writer(file)
    head=header[5:]
    writer.writerow(["Attr","Top 3 Highest","Top 3 Lowest"])
    for i in range(NUMBER_OF_ATTR):
        players_after_sort=sorted(to_compare_list,key=lambda x:x[5+i])
        top3H="Top 1: "+players_after_sort[-1][0]+"(Team: "+players_after_sort[-1][2]+")"+"\\n\\n"
            +"Top 2 from bottom: : "+players_after_sort[-2][0]+"(Team: "+players_after_sort[-2][2]+")"+"\\n\\n"
            +"Top 3: "+players_after_sort[-3][0]+"(Team: "+players_after_sort[-3][2]+")"

        top3L="Top 1 from bottom: "+players_after_sort[1][0]+"(Team: "+players_after_sort[1][2]+")"+"\\n\\n"
            +"Top 2 from bottom: : "+players_after_sort[2][0]+"(Team: "+players_after_sort[2][2]+")"+"\\n\\n"
            +"Top 3 from bottom: : "+players_after_sort[3][0]+"(Team: "+players_after_sort[3][2]+")"

        writer.writerow([head[i],top3H,top3L])

os.startfile(file_path_resultTop3)
```

-Cuối cùng mở file result3.csv ta có thể xem thông tin về kết quả

Attr	Top 3 Highest	Top 3 Lowest
Attr	Top 3 Highest	Top 3 Lowest
matches_played	Top 1: William Saliba(Team: Arsenal) Top 2: Vitaly Janelt(Team: Brentford) Top 3: Thomas Kaminski(Team: Luton Town)	Top 1 from bottom: Ademola Ola-Adebomi(Team: Crystal Palace) Top 2 from bottom: : Adriaen(Team: Liverpool) Top 3 from bottom: : Aidan Francis(Team: Luton Town)
starts	Top 1: William Saliba(Team: Arsenal) Top 2: Thomas Kaminski(Team: Luton Town) Top 3: Max Kilman(Team: Wolves)	Top 1 from bottom: Ademola Ola-Adebomi(Team: Crystal Palace) Top 2 from bottom: : Adriaen(Team: Liverpool) Top 3 from bottom: : Aidan Francis(Team: Luton Town)
minutes	Top 1: William Saliba(Team: Arsenal) Top 2: Max Kilman(Team: Wolves) Top 3: Jordan Pickford(Team: Everton)	Top 1 from bottom: Ademola Ola-Adebomi(Team: Crystal Palace) Top 2 from bottom: : Adriaen(Team: Liverpool) Top 3 from bottom: : Aidan Francis(Team: Luton Town)
non_penalty_goals	Top 1: Erling Haaland(Team: Manchester City) Top 2: Phil Foden(Team: Manchester City) Top 3: Ollie Watkins(Team: Aston Villa)	Top 1 from bottom: Aaron Hickey(Team: Brentford) Top 2 from bottom: : Aaron Ramsdale(Team: Arsenal) Top 3 from bottom: : Aaron Ramsey(Team: Burnley)
penalty_goals	Top 1: Cole Palmer(Team: Chelsea) Top 2: Erling Haaland(Team: Manchester City) Top 3: Bukayo Saka(Team: Arsenal)	Top 1 from bottom: Aaron Hickey(Team: Brentford) Top 2 from bottom: : Aaron Ramsdale(Team: Arsenal) Top 3 from bottom: : Aaron Ramsey(Team: Burnley)
assists	Top 1: Ollie Watkins(Team: Aston Villa) Top 2: Cole Palmer(Team: Chelsea) Top 3: Son Heung-min(Team: Tottenham)	Top 1 from bottom: Aaron Hickey(Team: Brentford) Top 2 from bottom: : Aaron Ramsdale(Team: Arsenal) Top 3 from bottom: : Aaron Ramsey(Team: Burnley)
yellow_cards	Top 1: Marcos Senes(Team: Bournemouth) Top 2: JoAto Palhinha(Team: Fulham) Top 3: Mario Lemina(Team: Wolves)	Top 1 from bottom: Adam Davies(Team: Sheffield Utd) Top 2 from bottom: : Ademola Ola-Adebomi(Team: Crystal Palace) Top 3 from bottom: : Adriaen(Team: Liverpool)
	Top 1: Yves Bissouma(Team: Tottenham)	Top 1 from bottom: Aaron Hickey(Team: Brentford)

c) 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

*Ý tưởng: xây dựng hàm để gom từng thuộc tính của các đối tượng Player vào 1 mảng

Vậy sẽ có 167 mảng tương ứng với 167 thuộc tính có thể so sánh, 1 mảng sẽ có độ dài bằng số lượng Player hiện có, 167 mảng này sẽ được quản lý bởi mảng all_attr

```
def get_attr_array(list_player):
    all_list_attr_of_player=[] #1 phần tử là tất cả các thuộc tính của Player

    for i in list_player:
        arr=row(i)
        all_list_attr_of_player.append(arr[5:])

    all_attr=[]
    for i in range(NUMBER_OF_ATTR): #1 Phần tử là 1 dãy cùng 1 thuộc tính của các Player
        for player_attr in all_list_attr_of_player:
            attr_value=player_attr[i]
            if attr_value!="N/a":
                all_attr[i].append(attr_value)

    return all_attr
```

Hàm def get_attr_array(list_player) sẽ thực hiện yêu cầu này bằng cách:

- Lấy 167 thuộc tính của Player lưu thành 1 mảng, mảng này được lưu vào all_list_attr_of_player
- Từ mảng all_list_attr_of_player, ta lấy giá trị đầu tiên của tất cả các mảng con trong all_list_attr_of_player lưu thành 1 mảng và thêm vào mảng all_attr
- Cứ thế ta lấy giá trị thứ hai, thứ 3,... cho đến hết và tổng cộng có 167 mảng tương ứng được thêm vào mảng all_attr
- Tất cả các giá trị không hợp lệ sẽ bị loại bỏ thay vì gán chúng bằng 0 để đảm bảo các giá trị trung bình, trung vị, độ lệch chuẩn không bị lệch quá nhiều

*Tìm trung vị, trung bình, độ lệch chuẩn cho tất cả các Player

```
#start all player
all_attr=get_attr_array(list_player)#1 Phần tử là 1 dãy cùng 1 thuộc tính của các Player,Độ dài của 1 phần tử là số lượng Player
mean_value_list=[0]*NUMBER_OF_ATTR
median_value_list=[0]*NUMBER_OF_ATTR
std_dev_list=[0]*NUMBER_OF_ATTR

for index,arr in enumerate(all_attr):
    mean_value_list[index]=statistics.mean(arr)
    median_value_list[index]=statistics.median(arr)
    std_dev_list[index]=statistics.stdev(arr)

writer = csv.writer(file)
writer.writerow(header2)

r=row2(0,"All",median_value_list,mean_value_list,std_dev_list)
writer.writerow(r)
# #end all player
```

- Lấy được mảng all_attr từ hàm ta mới xây dựng đồng thời khởi tạo 3 mảng mean_value_list, median_value_list, std_dev_list, mỗi phần tử là 1 thuộc tính của đối tượng

-Có thể tính các giá trị dựa vào thư viện statistics

-header2 là mảng các đầu đề của bảng được định nghĩa ở file title

```
# header2=[]
# for i in header[5:]:
#     header2.append("Median of "+i)
#     header2.append("Mean of "+i)
#     header2.append("Std of "+i)
# print(header2)

header2=[' ',' ','Median of matches_played', 'Mean of matches_played', 'Std of matches_played', 'Median of starts', 'Mean of starts', 'Std of starts',
'Median of minutes', 'Mean of minutes', 'Std of minutes', 'Median of non_penalty_goals', 'Mean of non_penalty_goals',
'Std of non_penalty_goals', 'Median of penalty_goals', 'Mean of penalty_goals', 'Std of penalty_goals', 'Median of assists',
'Mean of assists', 'Std of assists', 'Median of yellow_cards', 'Mean of yellow_cards', 'Std of yellow_cards', 'Median of red_cards',
'Mean of red_cards', 'Std of red_cards', 'Median of xG', 'Mean of xG', 'Std of xG', 'Median of npxG', 'Mean of npxG', 'Std of npxG',
'Median of xAG', 'Mean of xAG', 'Std of xAG', 'Median of PrGC', 'Mean of PrGC', 'Std of PrGC', 'Median of PrGP', 'Mean of PrGP',
'Std of PrGP', 'Median of PrGR', 'Mean of PrGR', 'Std of PrGR', 'Median of per90_Gls', 'Mean of per90_Gls', 'Std of per90_Gls',
'Median of per90_Ast', 'Mean of per90_Ast', 'Std of per90_Ast', 'Median of per90_G+A', 'Mean of per90_G+A', 'Std of per90_G+A',
'Median of per90_G-PK', 'Mean of per90_G-PK', 'Std of per90_G-PK', 'Median of per90_G+A-PK', 'Mean of per90_G+A-PK',
'Std of per90_G+A-PK', 'Median of per90_xG', 'Mean of per90_xG', 'Std of per90_xG', 'Median of per90_xAG', 'Mean of per90_xAG',
'Std of per90_xAG', 'Median of per90_xG+xAG', 'Mean of per90_xG+xAG', 'Std of per90_xG+xAG', 'Median of per90_npxG',
'Mean of per90_npxG', 'Std of per90_npxG', 'Median of per90_npxG+xAG', 'Mean of per90_npxG+xAG', 'Std of per90_npxG+xAG',
'Median of GA', 'Mean of GA', 'Std of GA', 'Median of GA90', 'Mean of GA90', 'Std of GA90', 'Median of SOTA', 'Mean of SOTA',
'Std of SOTA', 'Median of Saves', 'Mean of Saves', 'Std of Saves', 'Median of Save%', 'Mean of Save%', 'Std of Save%', 'Median of W',
'Mean of W', 'Std of W', 'Median of D', 'Mean of D', 'Std of D', 'Median of L', 'Mean of L', 'Std of L', 'Median of CS', 'Mean of CS',
'Std of CS', 'Median of CS%', 'Mean of CS%', 'Std of CS%', 'Median of PKatt', 'Mean of PKatt', 'Std of PKatt', 'Median of PKA',
'Mean of PKA', 'Std of PKA', 'Median of PKsv', 'Mean of PKsv', 'Std of PKsv', 'Median of PKm', 'Mean of PKm', 'Std of PKm',
'Median of GK_Save%', 'Mean of GK_Save%', 'Std of GK_Save%', 'Median of Gls', 'Mean of Gls', 'Std of Gls', 'Median of Sh',
'Mean of Sh', 'Std of Sh', 'Median of SoT', 'Mean of SoT', 'Std of SoT', 'Median of SoT%', 'Mean of SoT%', 'Std of SoT%',
'Median of Sh/90', 'Mean of Sh/90', 'Std of Sh/90', 'Median of SoT/90', 'Mean of SoT/90', 'Std of SoT/90', 'Median of G/Sh',
'Mean of G/Sh', 'Std of G/Sh', 'Median of G/SoT', 'Mean of G/SoT', 'Std of G/SoT', 'Median of Dist', 'Mean of Dist', 'Std of Dist',
'Median of FK', 'Mean of FK', 'Std of FK', 'Median of PK', 'Mean of PK', 'Std of PK', 'Median of PKatt', 'Mean of PKatt',
'Std of PKatt', 'Median of xG', 'Mean of xG', 'Std of xG', 'Median of npxG', 'Mean of npxG', 'Std of npxG', 'Median of npxG/Sh',
'Mean of npxG/Sh', 'Std of npxG/Sh', 'Median of G-xG', 'Mean of G-xG', 'Std of G-xG', 'Median of np:G-xG', 'Mean of np:G-xG',
'Std of np:G-xG', 'Median of Pass_Cmp', 'Mean of Pass_Cmp', 'Std of Pass_Cmp', 'Median of Pass_Att', 'Mean of Pass_Att',
'Std of Pass_Att', 'Median of Pass_Cmp%', 'Mean of Pass_Cmp%', 'Std of Pass_Cmp%', 'Median of TotDist', 'Mean of TotDist',
'Std of TotDist', 'Median of PreDist', 'Mean of PreDist', 'Std of PreDist', 'Median of Short_Cmp', 'Mean of Short_Cmp']
```

-row2 là hàm trả về mảng giá trị của các thuộc tính,nhận tham số là các mảng trung vị,trung bình, độ lệch chuẩn

```
def row2(stt,teamName,median_value_list,mean_value_list,std_dev_list):
    arr=[stt,teamName]
    for i in range(len(median_value_list)):
        arr.append(median_value_list[i])
        arr.append(mean_value_list[i])
        arr.append(std_dev_list[i])
    return arr
```

Ta gọi hàm row2 sau đó viết vào file csv

*Tìm trung vị, trung bình, độ lệch chuẩn cho tất cả các Player trong từng team

-Tương tự như tất cả các Player

```
#start team
for stt,squad in enumerate(list_squad):
    all_attr=get_attr_array(squad.players)#1 Phần tử là 1 dãy cũng 1 thuộc tính của các Player, độ dài của 1 phần tử là số lượng Player trong team

    mean_value_list=[0]*NUMBER_OF_ATTR
    median_value_list=[0]*NUMBER_OF_ATTR
    std_dev_list=[0]*NUMBER_OF_ATTR

    for index,arr in enumerate(all_attr):
        mean_value_list[index]=statistics.mean(arr)
        median_value_list[index]=statistics.median(arr)
        if len(arr)<2:
            std_dev_list[index]="N/a"
        else: std_dev_list[index]=statistics.stdev(arr)

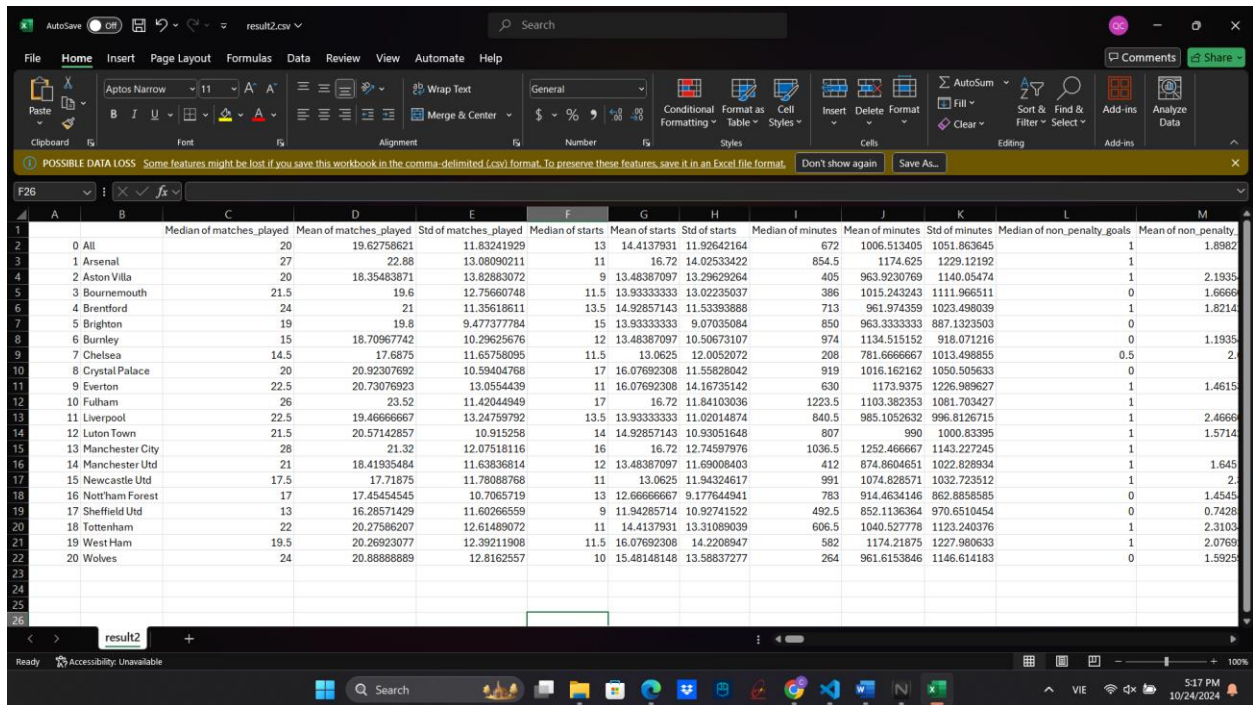
    writer = csv.writer(file)
    r=row2(stt+1,squad.name,median_value_list,mean_value_list,std_dev_list)
    writer.writerow(r)
#end team
```

Chú ý: tham số truyền vào hàm get_attr_array không phải tất cả Player mà chỉ là các Player có trong team đó thôi, có thêm điều kiện độ dài của mảng nhỏ hơn 2 do số lượng Player trong team quá ít nên không đủ số lượng để tính độ lệch chuẩn (1 đội nào đó chỉ có 1 thủ môn, nên các chỉ số của thủ môn chỉ có 1 nên không thể tính độ lệch chuẩn)

```
for index,arr in enumerate(all_attr):
    mean_value_list[index]=statistics.mean(arr)
    median_value_list[index]=statistics.median(arr)
    if len(arr)<2:
        std_dev_list[index]="N/a"
    else: std_dev_list[index]=statistics.stdev(arr)
```

Ta tính được trung vị, trung bình, độ lệch chuẩn cho tất cả 20 team sau đó lưu vào file tương tự như phần trên

Cuối cùng mở file và hoàn thành



	A	B	C	D	E	F	G	H	I	J	K	L	M
1			Median of matches played	Mean of matches played	Std of matches played	Median of starts	Mean of starts	Std of starts	Median of minutes	Mean of minutes	Std of minutes	Median of non_penalty_goals	Mean of non_penalty
2	0	All	20	19.62758621	11.83241929	13	14.4137931	11.92642164	672	1006.513405	1051.863645	1	1.8982
3	1	Arsenal	27	22.88	13.08090211	11	16.72	14.02533422	854.5	1174.625	1229.12192	1	
4	2	Aston Villa	20	18.35483871	13.82883072	9	13.48387097	13.29629264	405	963.9230769	1140.05474	1	2.1935
5	3	Bournemouth	21.5	19.6	12.75660748	11.5	13.93333333	13.02235037	386	1015.243243	1111.966511	0	1.6666
6	4	Brentford	24	21	11.35618611	13.5	14.92857143	11.53393888	713	961.974359	1023.498039	1	1.8214
7	5	Brighton	19	19.8	9.477377784	15	13.93333333	9.07035084	850	963.3333333	887.1323503	0	
8	6	Burnley	15	18.70967742	10.29625676	12	13.48387097	10.50673107	974	1134.515152	918.071216	0	1.1935
9	7	Chelsea	14.5	17.6875	11.65758095	11.5	13.0625	12.0052072	208	781.6666667	1013.498855	0.5	2.
10	8	Crystal Palace	20	20.92307692	10.59404768	17	16.07692308	11.55828042	919	1016.162162	1050.505633	0	
11	9	Everton	22.5	20.73076923	13.0554439	11	16.07692308	14.16735142	630	1173.9375	1226.989627	1	1.4615
12	10	Fulham	26	23.52	11.42044949	17	16.72	11.84103036	1223.5	1103.382353	1081.703427	1	
13	11	Liverpool	22.5	19.46666667	13.24759792	13.5	13.93333333	11.02014874	840.5	985.1052632	996.8126715	1	2.4666
14	12	Luton Town	21.5	20.57142857	10.915258	14	14.92857143	10.93051648	807	990	1000.83395	1	1.5714
15	13	Manchester City	28	21.32	12.07518116	16	16.72	12.74597976	1036.5	1252.466667	1143.227245	1	
16	14	Manchester Utd	21	18.41935484	11.63836814	12	13.48387097	11.69008403	412	874.8604651	1022.828934	1	1.645
17	15	Newcastle Utd	17.5	17.71875	11.78088768	11	13.0625	11.94324617	991	1074.828571	1032.723512	1	2.
18	16	Nottingham Forest	17	17.45454545	10.7065719	13	12.66666667	9.177644941	783	914.4634146	862.8858585	0	1.4545
19	17	Sheffild Utd	13	16.28571429	11.60266559	9	11.94285714	10.92741522	492.5	852.1136364	970.6510454	0	0.7428
20	18	Tottenham	22	20.27586207	12.61489072	11	14.4137931	13.31089039	606.5	1040.527778	1123.240376	1	2.3103
21	19	West Ham	19.5	20.26923077	12.39211908	11.5	16.07692308	14.2208947	582	1174.21875	1227.980633	1	2.0769
22	20	Wolves	24	20.88888889	12.8162557	10	15.48148148	13.58637277	264	961.6153846	1146.614183	0	1.5925

d) Vẽ histogram

Ta tận dụng hàm `get_attr_array` của phần trên để tập hợp các dữ liệu cùng 1 loại phục vụ cho việc vẽ histogram

Do số lượng histogram khá lớn, ta lưu vào 1 file pdf để tiện theo dõi

-Vẽ histogram cho tất cả player

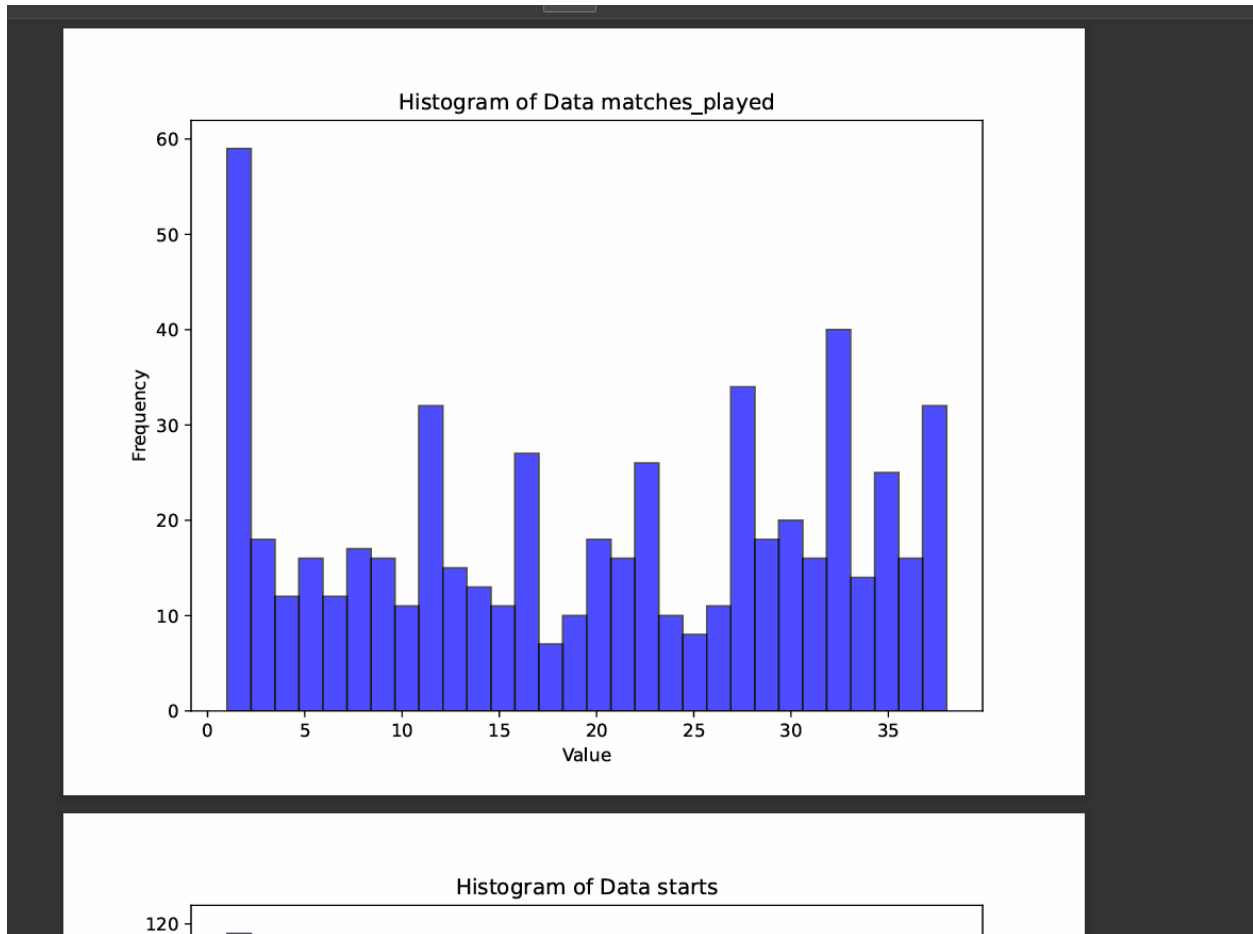
```
from common import header
with PdfPages('histogramsAllPlayer.pdf') as pdf:
    # Vẽ histogram cho dữ liệu 1
    #start all player
    all_attr=get_attr_array(list_player)#1 phần tử là 1 thuộc tính của tất cả player, độ dài 1 phần tử là số lượng player




















    for index,arr in enumerate(all_attr):
        plt.figure(figsize=(8, 6))
        plt.hist(arr, bins=30, color='blue', alpha=0.7, edgecolor='black')
        plt.title(f'Histogram of Data {header[5+index]}')
        plt.xlabel('Value')
        plt.ylabel('Frequency')
        pdf.savefig() # Lưu biểu đồ vào tệp PDF
        plt.close() # Đóng biểu đồ
    print("Các biểu đồ đã được lưu vào 'histogramsAllPlayer.pdf'")
```

- Vẽ histogram cho từng đội

```
# Vẽ histogram cho team
for stt,squad in enumerate(list_squad):
    with PdfPages(f'histogramsSquad{squad.name.replace(" ","")}.pdf') as pdf:
        all_attr=get_attr_array(squad.players)#1 phần tử là 1 thuộc tính của tất cả player, độ dài 1 phần tử là số lượng player
        for index,arr in enumerate(all_attr):
            plt.figure(figsize=(8, 6))
            plt.hist(arr, bins=30, color='blue', alpha=0.7, edgecolor='black')
            plt.title(f'Histogram of Data {header[5+index]}')
            plt.xlabel('Value')
            plt.ylabel('Frequency')
            pdf.savefig() # Lưu biểu đồ vào tệp PDF
            plt.close() # Đóng biểu đồ
        print(f"Các biểu đồ đã được lưu vào 'histogramsSquad{squad.name.replace(" ","")}.pdf'")
```

Ta thu được 1 file chứa biểu đồ histogram của tất cả các cầu thủ và các file chứa histogram của từng đội tuyển như hình bên dưới



	histogramsAllPlayer.pdf	10/21/2024 3:36 PM	Microsoft Edge PD...	284 KB
	histogramsSquadArsenal.pdf	10/21/2024 3:38 PM	Microsoft Edge PD...	266 KB
	histogramsSquadAstonVilla.pdf	10/21/2024 3:39 PM	Microsoft Edge PD...	266 KB
	histogramsSquadBournemouth.pdf	10/21/2024 3:40 PM	Microsoft Edge PD...	266 KB
	histogramsSquadBrentford.pdf	10/21/2024 3:41 PM	Microsoft Edge PD...	267 KB
	histogramsSquadBrighton.pdf	10/21/2024 3:42 PM	Microsoft Edge PD...	267 KB
	histogramsSquadBurnley.pdf	10/21/2024 3:42 PM	Microsoft Edge PD...	266 KB
	histogramsSquadChelsea.pdf	10/21/2024 3:43 PM	Microsoft Edge PD...	267 KB
	histogramsSquadCrystalPalace.pdf	10/21/2024 3:44 PM	Microsoft Edge PD...	266 KB
	histogramsSquadEverton.pdf	10/21/2024 3:45 PM	Microsoft Edge PD...	264 KB
	histogramsSquadFulham.pdf	10/21/2024 3:47 PM	Microsoft Edge PD...	265 KB
	histogramsSquadLiverpool.pdf	10/21/2024 3:48 PM	Microsoft Edge PD...	266 KB
	histogramsSquadLutonTown.pdf	10/21/2024 3:50 PM	Microsoft Edge PD...	266 KB
	histogramsSquadManchesterCity.pdf	10/21/2024 3:51 PM	Microsoft Edge PD...	266 KB
	histogramsSquadManchesterUtd.pdf	10/21/2024 3:52 PM	Microsoft Edge PD...	265 KB
	histogramsSquadNewcastleUtd.pdf	10/21/2024 3:53 PM	Microsoft Edge PD...	266 KB
	histogramsSquadNott'hamForest.pdf	10/21/2024 3:54 PM	Microsoft Edge PD...	267 KB
	histogramsSquadSheffieldUtd.pdf	10/21/2024 4:01 PM	Microsoft Edge PD...	265 KB
	histogramsSquadTottenham.pdf	10/21/2024 4:01 PM	Microsoft Edge PD...	265 KB

****Chú ý:** Các biểu đồ histogram đã được vẽ sẵn bằng hàm drawHistogram() tạo ra các file pdf nên không nhất thiết phải chạy lại hàm này nữa nên hàm này sẽ bị comment

```
findtop3()
findMeanMedianStddev()
# drawHistogram()
```

e) Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số

Ý tưởng: xây dựng 2 mảng, 1 mảng chứa chỉ số cao nhất ở mỗi thuộc tính, 1 mảng chứa tên đội có giá trị thuộc tính cao nhất ứng với chỉ số của mảng đầu tiên

```
biggest_attr_value=[-1e9]*NUMBER_OF_ATTR
best_team_in_one_attr=[""]*NUMBER_OF_ATTR
```


Như vậy, cả 2 mảng sẽ có độ dài là 167 tương ứng với 167 thuộc tính có thể so sánh được

Hàm rowsquad được định nghĩa ở file title, nhận tham số là 1 đối tượng Squad, trả về mảng là giá trị các thuộc tính của đối tượng đó

```
def rowsquad(squad):
    return [
        squad.players, squad.name, squad.numberOfPlayer, squad.poss, squad.age,
        # Playing Time
        squad.playing_time["matches_played"], squad.playing_time["starts"], squad.playing_time["minutes"],
        # Performance
        squad.performance["non_penalty_goals"], squad.performance["penalty_goals"], squad.performance["assists"],
        squad.performance["yellow_cards"], squad.performance["red_cards"],
        # Expected
        squad.expected["xG"], squad.expected["npxG"], squad.expected["xAG"],
        # Progression
        squad.progression["PrgC"], squad.progression["PrgP"], squad.progression["PrgR"],
        # Per 90 minutes
        squad.per_90["Gls"], squad.per_90["Ast"], squad.per_90["G+A"], squad.per_90["G-PK"],
        squad.per_90["G+A-PK"], squad.per_90["xG"], squad.per_90["xAG"], squad.per_90["xG+xAG"],
        squad.per_90["npxG"], squad.per_90["npxG+xAG"],
        # Goalkeeping Performance
        squad.goalkeeping["Performance"]["GA"], squad.goalkeeping["Performance"]["GA90"],
        squad.goalkeeping["Performance"]["SoTA"], squad.goalkeeping["Performance"]["Saves"],
        squad.goalkeeping["Performance"]["Save%"], squad.goalkeeping["Performance"]["W"],
        squad.goalkeeping["Performance"]["D"], squad.goalkeeping["Performance"]["L"],
        squad.goalkeeping["Performance"]["CS"], squad.goalkeeping["Performance"]["CS%"],
        # Goalkeeping Penalty Kicks
        squad.goalkeeping["Penalty Kicks"]["PKatt"], squad.goalkeeping["Penalty Kicks"]["PKA"],
        squad.goalkeeping["Penalty Kicks"]["PKsv"], squad.goalkeeping["Penalty Kicks"]["PKm"],
        squad.goalkeeping["Penalty Kicks"]["Save%"],
        # Shooting Standard
        squad.shooting["Standard"]["Gls"], squad.shooting["Standard"]["Sh"],
```

Thực hiện lặp qua từng đội, sau đó lặp qua từng thuộc tính của đội nếu thuộc tính hiện tại của đội này lớn hơn giá trị trong mảng biggest_attr_value tương ứng thì ta thực hiện thay đổi giá trị trong mảng biggest_attr_value đồng thời cũng thay đổi tên của đội có điểm cao nhất tại vị trí đó luôn

```
best_team_in_one_attr = ""
for i in list_squad:
    arr = rowsquad(i)[5:]
    for index, value in enumerate(arr):
        if value != "N/a" and value > biggest_attr_value[index]:
            biggest_attr_value[index] = value
            best_team_in_one_attr[index] = i.name

        elif value == "N/a" and biggest_attr_value[index] == 0 and best_team_in_one_attr[index] == "":
            biggest_attr_value[index] = 0
            best_team_in_one_attr[index] = i.name
```

Thêm 1 trường hợp xử lý nếu chưa có giá trị cao nhất nào được thiết lập trong mảng thì đội hiện tại sẽ

Cuối cùng thực hiện in kết quả

```
for index,i in enumerate(best_team_in_one_attr):
    print(header[5+index]+": ",i)
```

*Đánh giá đội có phong độ tốt nhất mùa giải

Ta thực hiện đếm số lần mà đội đó xuất hiện trong mảng best_team_in_one_attr

Đội nào xuất hiện nhiều nhất thì có nghĩa là đội đó dẫn đầu nhiều mặt chỉ số nên có phong độ rất cao

```
#đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024
from collections import Counter
squad_ranking=Counter(best_team_in_one_attr)
print("Best Squad in Cometition is: ",(squad_ranking.most_common(1)[0][0]))
#End find best squad*****
```

Ở đây ta dùng Counter để đếm số lần mỗi đội xuất hiện, ta tìm được tên của đội có phong độ tốt nhất

```
Best Squad in Cometition is: Manchester City
```

Kết quả cho thấy Manchester City có mặt ở nhiều chỉ số cao nhất, thực tế cũng chứng minh rằng đội bóng này là đội bóng vô địch trong mùa giải

3. Bài tập 3

a) Các thư viện cần thiết

Các thư viện cần dùng

```
import pickle
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import PowerTransformer
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from radarChartPlot import drawRadarChart
from title import header,row
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

from object import Player_Manager
```

```
# Đọc các đối tượng từ file
file_path1 = os.path.join(os.path.dirname(__file__), "players.pkl")
list_player=[]
with open(file_path1, "rb") as file:
    list_player = pickle.load(file)

print("Load data success!")

list_player=list(filter(lambda p:p.playing_time["minutes"]>90,list_player))
print(len(list_player))
```

Ta sử dụng mảng đối tượng Player đã được lưu ở bài 1

Mảng list_player chứa những cầu thủ chơi trên 90 phút

Đồng thời lấy luôn mảng giá trị trung bình đã được tính toán ở phần bài 2

```
#lay tu ket qua cua bai 2
mean_value_list=[19.627586206896552, 14.413793103448276, 1006.5134048257372, 1.8982758620689655, 0.16551724137931034,
1.4810344827586206, 2.85, 0.1, 2.0760344827586206, 1.9296551724137931, 1.4937931034482759, 24.651724137931033,
50.58448275862069, 50.06206896551724, 0.12525862068965518, 0.09162068965517242, 0.2168103448275862,
0.1181551724137931, 0.20963793103448275, 0.14498275862068966, 0.10070689655172414, 0.2458448275862069,
0.13843103448275862, 0.23946551724137932, 31.15, 2.24525, 91.3, 61.175, 65.44, 7.45, 4.1, 7.45, 3.925,
19.61578947368421, 2.675, 2.4, 0.2, 0.075, 9.720689655172414, 2.0637931034482757, 17.9, 6.117241379310345,
31.024516129032257, 1.354551724137931, 0.42579310344827587, 0.09219354838709677, 0.2941293532338308,
16.532043010752687, 0.4879310344827586, 0.16551724137931034, 0.18448275862068966, 2.0760344827586206,
1.9296551724137931, 0.10139784946236559, -0.012241379310344832, -0.03137931034482759, 541.5086206896551,
670.7896551724137, 78.10578947368421, 9199.26551724138, 3251.863793103448, 260.3965517241379, 291.3724137931035,
87.68645276292335, 220.91551724137932, 254.8448275862069, 83.14856115107914, 46.139655172413796, 88.82241379310345,
54.3527724665392, 1.4810344827586206, 1.4937931034482759, 1.3217241379310345, -0.012758620689655177,
13.587931034482759, 40.360344827586204, 11.686206896551724, 2.4362068965517243, 50.58448275862069,
608.8534482758621, 59.43793103448276, 16.103448275862068, 2.2913793103448277, 3.463793103448276, 22.686206896551724,
23.23448275862069, 7.086206896551724, 3.462068965517241, 2.070689655172414, 0.1706896551724138, 541.5086206896551,
2.4982758620689656, 12.579310344827586, 31.81896551724138, 2.352051724137931, 23.294827586206896, 2.646551724137931,
1.856896551724138, 2.0689655172413794, 1.3275862068965518, 0.6241379310344828, 3.5637931034482757,
0.23093103448275862, 2.5068965517241377, 0.22413793103448276, 0.25862068965517243, 0.2810344827586207,
0.22413793103448276, 0.06896551724137931, 23.137931034482758, 13.575862068965517, 11.21896551724138,
8.648275862068965, 3.2706896551724136, 10.74655172413793, 21.936206896551724, 46.83616600790514, 11.189655172413794,
16.04310344827586, 5.106896551724138, 10.936206896551724, 10.875862068965517, 34.01379310344828, 26.867241379310343,
0.5672413793103448, 818.9931034482759, 91.19137931034483, 268.66896551724136, 350.03275862068966, 207.90862068965518,
34.34827586206897, 818.8086206896552, 24.548275862068966, 11.189655172413794, 46.43340163934426, 10.74655172413793,
42.50553278688525, 468.73275862068965, 2477.746551724138, 1273.6155172413794, 24.651724137931033, 17.386206896551723,
7.85, 18.606896551724137, 12.391379310344828, 536.2344827586207, 50.06206896551724, 11.206434316353887,
79.69574036511156, 7.116621983914209, 4.053619302949062, 19.389898989898988, 5.081769436997319, 1.3616494845360825,
23.52920962199313, 23.457044673539517, 22.307241379310344, 22.231896551724137, 14.493103448275862, 13.98448275862069,
2.4982758620689656, 22.686206896551724, 0.08448275862068966, 63.26379310344828, 17.5, 17.5, 47.19566037735849]
```

b) Lọc và chuẩn hóa dữ liệu

-Ta bắt đầu bằng cách lọc các thuộc tính của từng đối tượng Player rồi lưu vào 1 mảng, tại đây, các giá trị không hợp lệ của đối tượng ta sẽ thay chúng bằng giá trị trung bình của các thuộc tính dựa trên mảng các thuộc tính đã khai báo ở trên

-Các thuộc tính có giá trị âm sẽ loại bỏ(chỉ cần 1 thuộc tính nào đó của 1 cầu thủ có giá trị âm, thì giá trị đó sẽ bị loại bỏ không đưa vào vẽ biểu đồ bởi nó chủ yếu là các giá trị tính bằng 1 hay nhiều giá trị thuộc tính khác (Ví dụ: per90_G+A tính bằng tổng per90_Gls và per90_Ast) nên chúng không quan trọng

```

def normalizedata():
    players_attr_list=[] #1phan tu la tat ca cac thuoc tinh cua player
    deleteAttr=[]

    for i in list_player:
        arr=row(i)[5:]
        #thay cac gia tri N/a ve gia tri trung binh cua toan bo player
        for i in range(len(arr)):
            if arr[i]=="N/a": arr[i]=mean_value_list[i]
            elif arr[i]<0:
                deleteAttr.append(i)
        players_attr_list.append(arr)

    non_negative_data=[]
    for player in players_attr_list:
        arr=[]
        for index,value in enumerate(player):
            if index not in deleteAttr:
                arr.append(value)
        non_negative_data.append(arr)

    #chuan hoa du lieu
    data = np.array(non_negative_data)
    data_normalized = np.log(data+(1e-18))
    scaler = MinMaxScaler()
    data_normalized = scaler.fit_transform(data_normalized)

    return data_normalized

```

-Hàm normalizedata thực hiện điều này với mảng players_attr_list lưu các thuộc tính của cầu thủ sau khi sửa các giá trị N/a, mảng AttrToDelete lưu các thuộc tính sẽ bị xóa vì thuộc tính đó có cầu thủ có giá trị nhỏ hơn 0

-Các giá trị sẽ được lọc lại một lần nữa cuối cùng ra được mảng non_negative_data, mảng này sẽ được đem đi chuẩn hóa làm dữ liệu cho biểu đồ

-Dữ liệu được chuẩn hóa 2 lần

+ Lần đầu tiên là chuẩn hóa bằng log do dữ liệu có độ lệch tương đối lớn(số phút có thể lên đến hàng nghìn nhưng số trận chỉ đến hàng chục,số bàn thắng cũng chỉ đến hàng chục, 1 số dữ liệu khác chỉ đến hàng đơn vị), ta cộng thêm 1 số rất nhỏ(1e-18) do dữ liệu có chứa giá trị 0

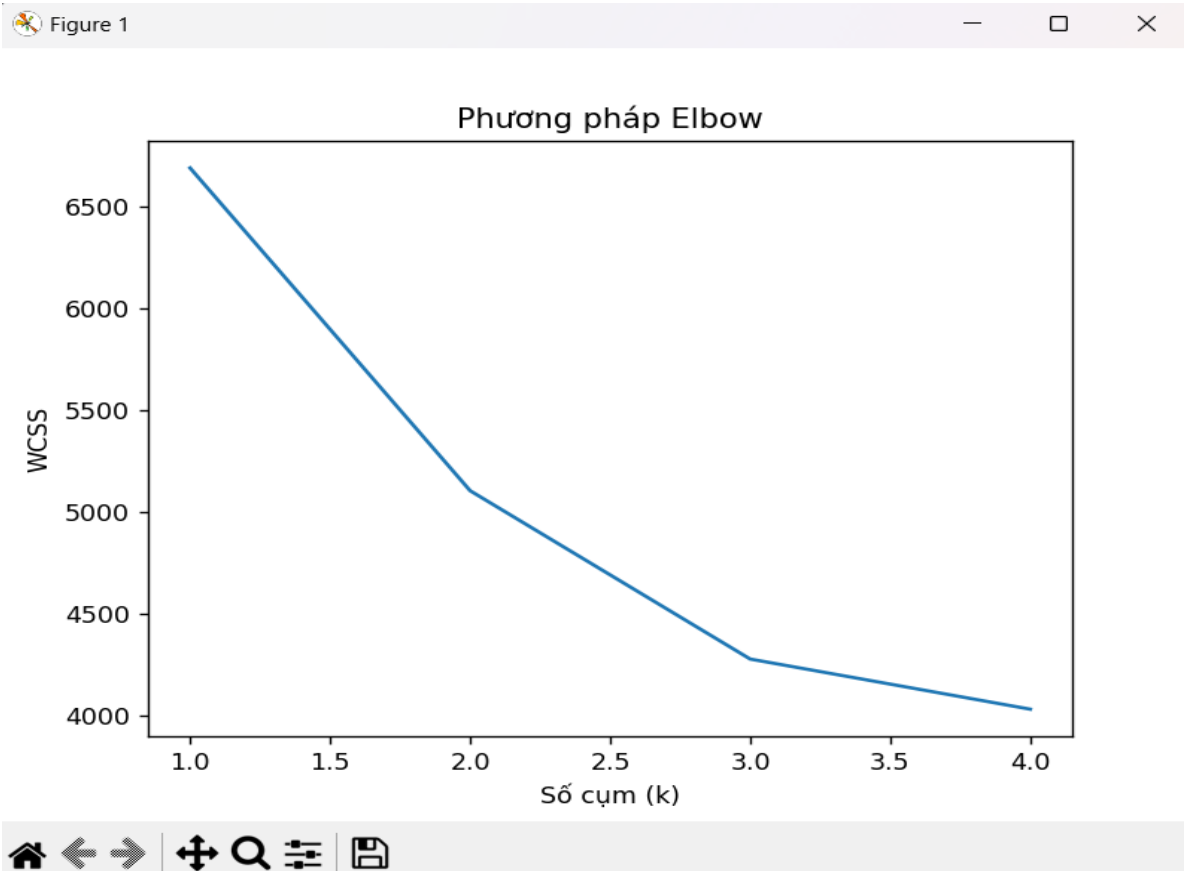
+ Lần thứ 2 là chuẩn hóa bằng thuật toán MinMaxScaler, thuật toán đưa các giá trị trong dữ liệu về khoảng 0-1

Hàm `normalizedata` trả về dữ liệu sau khi xử lý, chuẩn hóa cho việc vẽ sơ đồ

c) Tìm số điểm hội tụ cho thuật toán Kmean

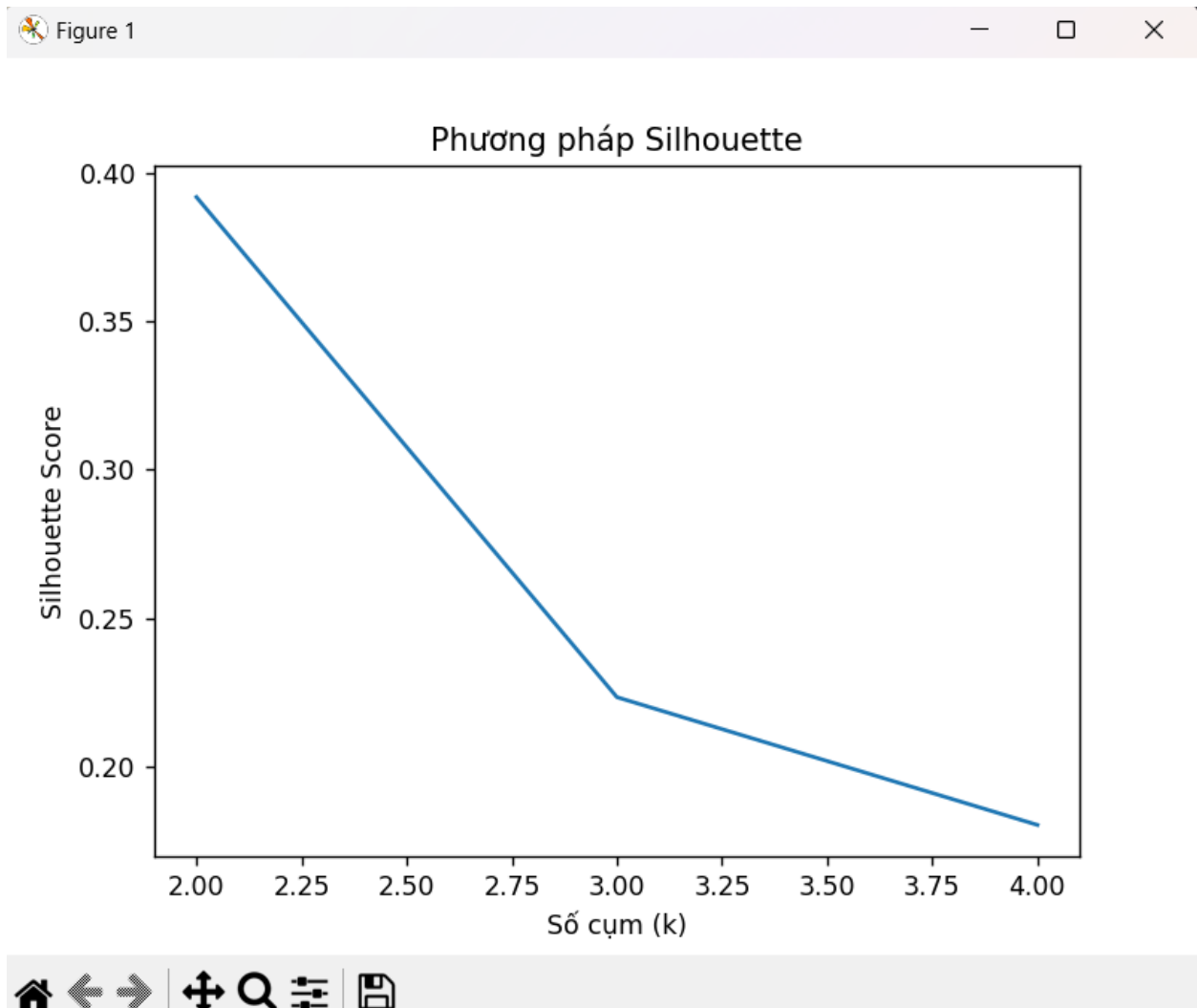
Ta có thể thực hiện đánh giá bằng phương pháp Elbow, số điểm phân cụm được đánh giá tại nơi mà ta thấy rõ điểm khuỷu tay trên sơ đồ

```
def Elbow_solution(data):  
    wcss = []  
    for k in range(1, 5):  
        kmeans = KMeans(n_clusters=k, random_state=10)  
        kmeans.fit(data)  
        wcss.append(kmeans.inertia_) # inertia_ là tổng WCSS  
  
    # Vẽ đồ thị Elbow  
    plt.plot(range(1, 5), wcss)  
    plt.title('Phương pháp Elbow')  
    plt.xlabel('Số cụm (k)')  
    plt.ylabel('WCSS')  
    plt.show()
```



Biểu đồ cho thấy gấp khúc ở đoạn 2 và 3

Ta có thể tiếp tục thử với phương pháp Silhouette



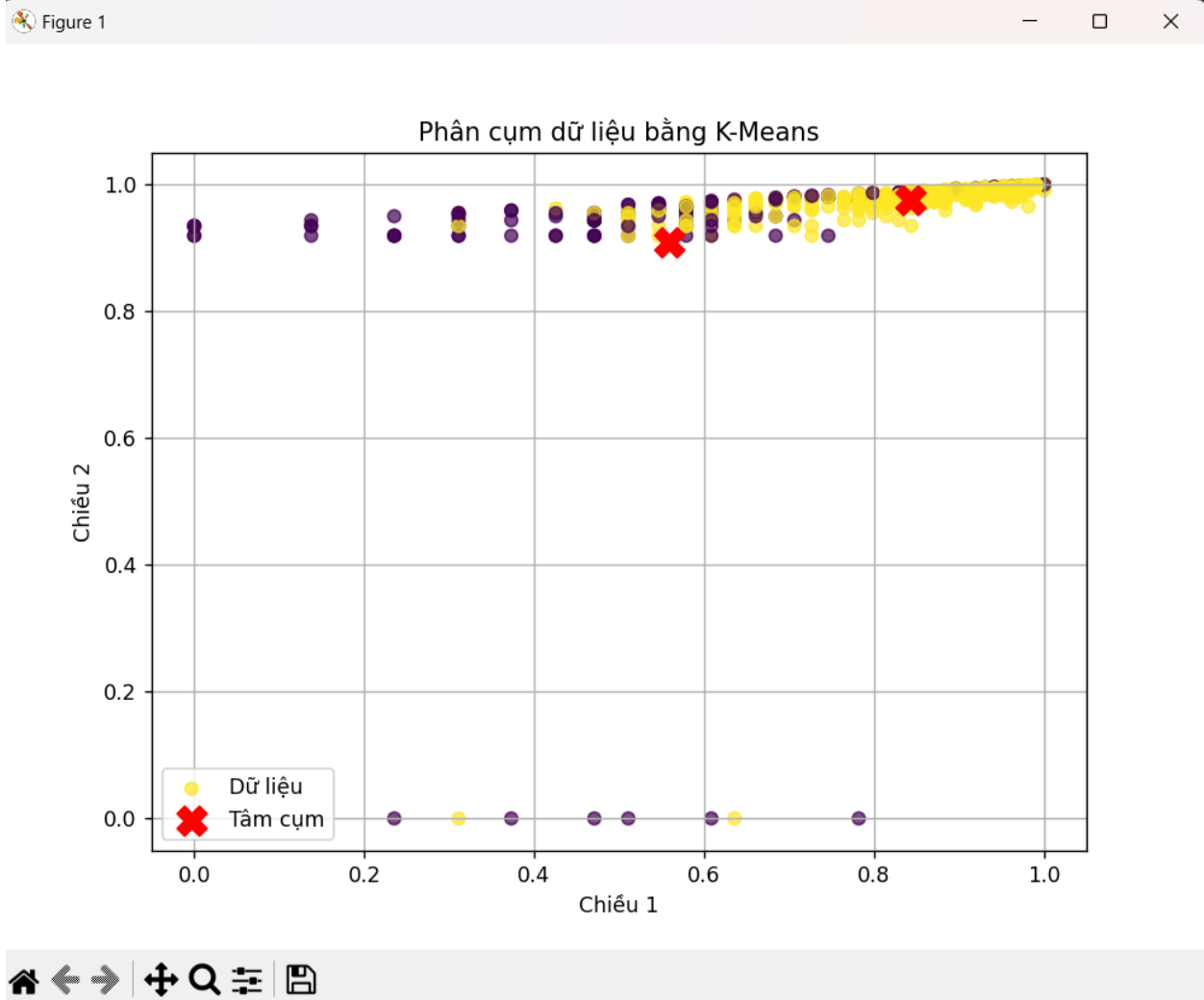
Dựa vào sơ đồ ta thấy điểm tại vị trí 2 đạt điểm cao nhất cho thấy sự tối ưu cao nhất nếu chia cụm thành 2 phần

*Nhận xét: Dựa vào số phần phân cụm đã các định chia cầu thủ thành 2 nhóm, ta thấy rõ được sự hợp lý khi 1 phần là cầu thủ kém nổi bật, các cầu thủ còn lại là các cầu thủ có thành tích và kỹ năng vượt trội hơn các cầu thủ còn lại

d) Phân cụm Kmean

-Thực hiện vẽ biểu đồ thông qua hàm drawKmean nhận 2 tham số là dữ liệu và số cụm muốn phân

```
def drawKmean(n,data_normalized):  
    kmeans = KMeans(n_clusters=n, random_state=50)  
    kmeans.fit(data_normalized)  
  
    # Lưu trữ nhãn cụm  
    labels = kmeans.labels_  
    # Lấy tọa độ của các tâm cụm  
    centroids = kmeans.cluster_centers_  
  
    # Vẽ dữ liệu và các tâm cụm  
    plt.figure(figsize=(8, 6))  
    plt.scatter(data_normalized[:, 0], data_normalized[:, 1], c=labels, cmap='viridis', marker='o', Label='Dữ liệu', alpha=0.7)  
    plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=200, marker='X', Label='Tâm cụm')  
    plt.title('Phân cụm dữ liệu bằng K-Means')  
    plt.xlabel('Chiều 1')  
    plt.ylabel('Chiều 2')  
    plt.legend()  
    plt.grid(True)  
    plt.show()
```



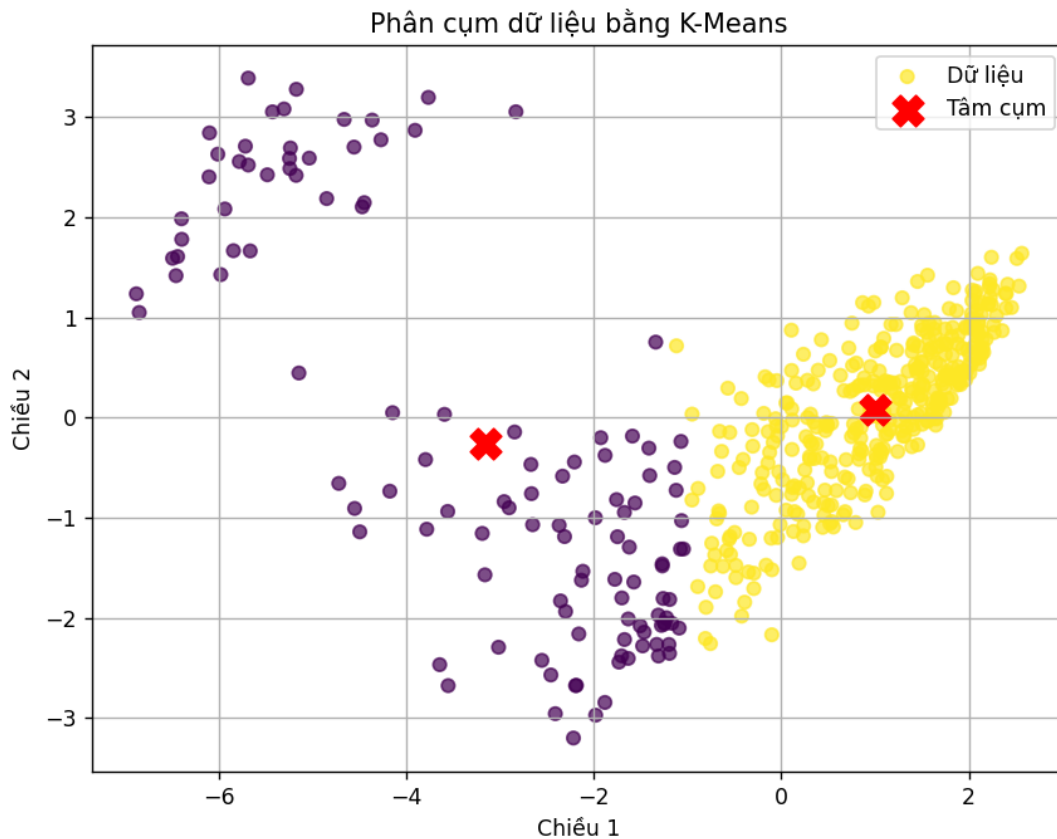
Phân cụm bằng Kmean

-Ta chưa thể thấy được sự rõ ràng trong phân cụm do dữ liệu đầu vào có rất nhiều chiều(167 chiều tương ứng với 167 thuộc tính)

e) Sử dụng phương pháp giảm chiều của dữ liệu bằng thuật toán PCA

```
# Áp dụng PCA
pca = PCA(n_components=2)
dataPCA = pca.fit_transform(data normalize)
drawKmean(2, dataPCA)
# drawChart()
```

Figure 1



g) So sánh 2 cầu thủ với radarChart

```
def ComparePlayer():
    player_manager=Player_Manager()
    player_manager.list_player=list_player
    print("So sánh 2 cầu thủ")
    player1=None
    player2=None
    while(True):
        name1=input("Nhập tên cầu thủ thứ nhất: ")
        squad1=input("Nhập tên đội của cầu thủ thứ nhất: ")
        player1=player_manager.findPlayerByNameandTeam(name1,squad1)
        if player1==None:
            print("Cầu thủ không tồn tại")
        else: break

    while(True):
        name2=input("Nhập tên cầu thủ thứ hai: ")
        squad2=input("Nhập tên đội của cầu thủ thứ hai: ")
        player2=player_manager.findPlayerByNameandTeam(name2,squad2)
        if player2==None:
            print("Cầu thủ không tồn tại")
        else: break

    #RadarChart
```

Ở đây ta lấy thông tin về tên của cầu thủ cùng với đội bóng của cầu thủ đó thi đấu, nếu thông tin không hợp lệ, ta yêu cầu nhập lại

Do thuộc tính của các cầu thủ tương đối phức tạp để nhớ cũng như nhập đúng, ta đặt cho mỗi thuộc tính 1 id và yêu cầu người nhập liệu nhập id vào thay vì tên của thuộc tính

Khi nhập thuộc tính, dữ liệu nhập được đặt trong vòng lặp vô hạn cho đến khi nhập đúng id thì thôi, các id được in ra trước để người dùng có thể theo dõi và nhập khi cần

```
matches_played : Id= 0
starts : Id= 1
minutes : Id= 2
non_penalty_goals : Id= 3
penalty_goals : Id= 4
```

```

#RadaChart
attr_list_to_draw=[]
attr_quantity_to_draw=0
while(True):
    try:
        attr_quantity_to_draw=int(input("Nhập số thuộc tính cần so sánh "))
        if attr_quantity_to_draw>167 or attr_quantity_to_draw<1: raise ValueError
        break
    except ValueError:
        print("Số lượng thuộc tính phải nhỏ hơn hoặc bằng 167 và lớn hơn 0")
for index,attr_name in enumerate(header[5:]):
    print(attr_name," : Id= ",index)

print("Nhập các id của thuộc tính cần so sánh")
for _ in range(attr_quantity_to_draw):
    n=0
    while(True):
        try:
            n=int(input("Id thuộc tính: "))
            if n>167 or n<0: raise ValueError
            break
        except ValueError:
            print("Id không hợp lệ")

    attr_list_to_draw.append(n)
# attr_list_to_draw=[0,1,11,12,13,47,67,68,69,71]
drawRadarChart(player1,player2,attr_list_to_draw)

```

Kết quả thu được từ dữ liệu đầu vào là 2 cầu thủ player1 và player2 cùng với đó là dãy các id thuộc tính cần so sánh được dùng làm tham số cho hàm drawRadarChart

* drawRadarChart được định nghĩa trong file radarChartplot đã nói ở phần đầu tiên, file này chỉ chứa duy nhất 1 hàm dùng để vẽ radarchart với tham số đầu vào là 2 đối tượng Player và mảng id thuộc tính cần so sánh

- Trong bước tiền xử lý, ta sẽ xử lý bằng cách lấy ra giá trị của các thuộc tính của đối tượng Player p1 và p2, tiếp theo lấy giá trị của các id thuộc tính có trong mảng tham số đầu, với những giá trị không hợp lệ, ta đặt nó bằng 0, kết quả cuối cùng là mảng values_player1 và values_player2 là 2 mảng chứa giá trị của các thuộc tính yêu cầu.

```

import numpy as np
import matplotlib.pyplot as plt

from title import row
from title import header

def drawRadarChart(p1, p2, arrOfAttr):
    # Lấy các nhãn tương ứng với các chỉ số
    head = header[5:]
    labels = [head[i] for i in arrOfAttr]

    # Lấy các giá trị chỉ số cho 2 cầu thủ
    values_player1 = []
    values_player2 = []
    attr_p1 = row(p1)[5:]
    attr_p2 = row(p2)[5:]

    for i in arrOfAttr:
        # Thay 'N/a' bằng giá trị 0
        value_p1 = 0 if attr_p1[i] == 'N/a' else float(attr_p1[i])
        value_p2 = 0 if attr_p2[i] == 'N/a' else float(attr_p2[i])
        values_player1.append(value_p1)
        values_player2.append(value_p2)

    # In ra để kiểm tra
    print("Values Player 1:", values_player1)
    print("Values Player 2:", values_player2)

```

Hoàn thành việc lọc dữ liệu ta tiến hành vẽ biểu đồ

```

# Tính toán góc cho các trục radar chart
angles = np.linspace(0, 2 * np.pi, len(labels), endpoint=False).tolist()

# Đảm bảo radar chart khép kín
values_player1 += values_player1[:1]
values_player2 += values_player2[:1]
angles += angles[:1]

# Tạo hình radar
fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))

# Vẽ radar cho cầu thủ 1
ax.fill(angles, values_player1, color='blue', alpha=0.25)
ax.plot(angles, values_player1, color='blue', linewidth=2, label=p1.name)

# Vẽ radar cho cầu thủ 2
ax.fill(angles, values_player2, color='red', alpha=0.25)
ax.plot(angles, values_player2, color='red', linewidth=2, label=p2.name)

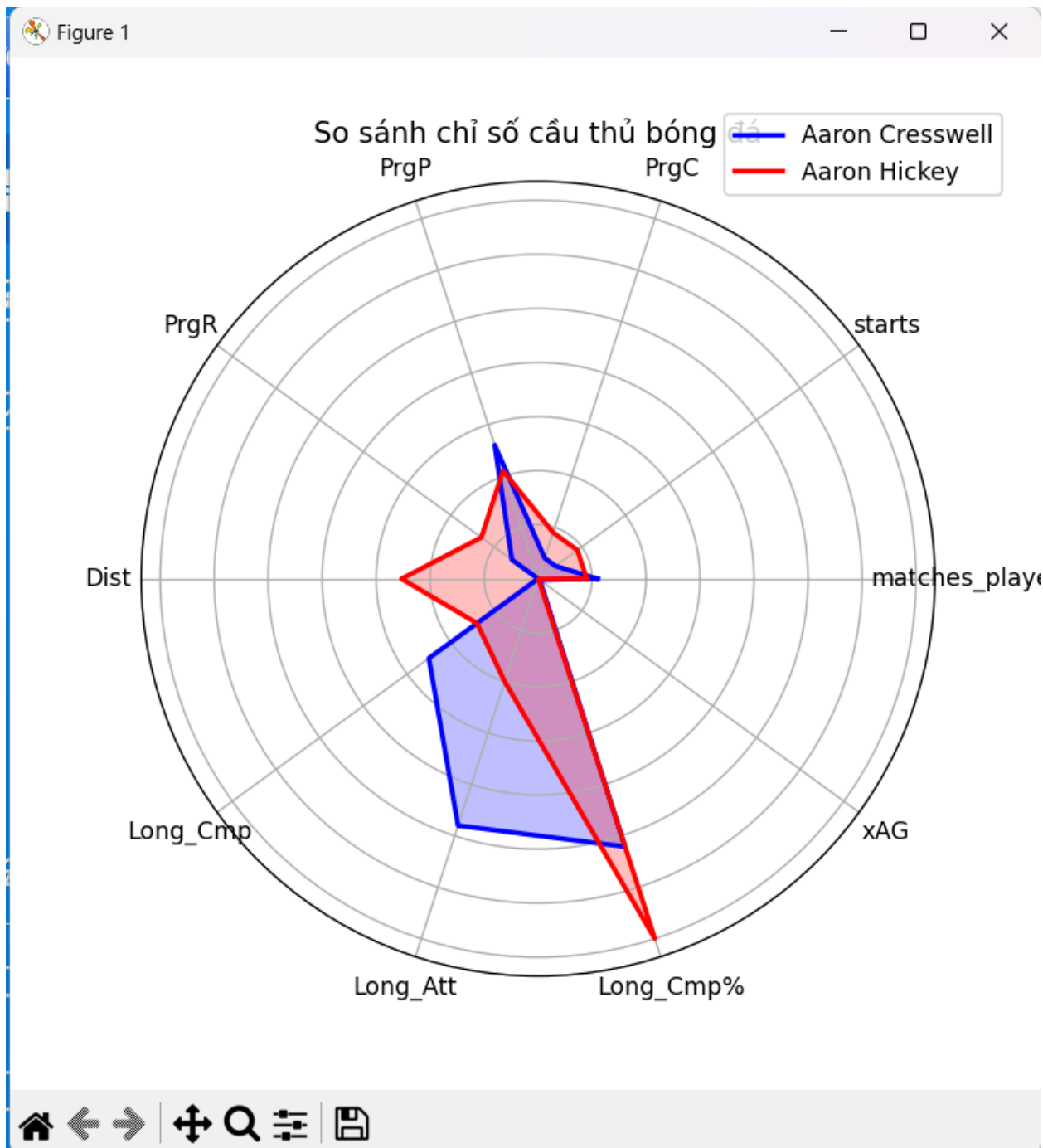
# Thiết lập nhãn cho các trục và bỏ nhãn giá trị trên trục y
ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)

# Thêm tiêu đề và chú thích
plt.title('So sánh chỉ số cầu thủ bóng đá')
plt.legend(loc='upper right', bbox_to_anchor=(1.1, 1.1))
plt.show()

```

****Demo kết quả:**

So sánh 2 cầu thủ Aaron Cresswell và Aaron Hickey ở các chỉ số có id 0,1,11,12,13,47,67,68,69,71 ta thu được biểu đồ như hình dưới



4. Bài tập 4

a) Update file object

- Do làm việc với 1 url khác, ta phải xây dựng 1 class đặc trưng cho việc chuyển nhượng cầu thủ

Do vậy, ta thêm vào trong file object.py một lớp nữa đặt tên là Tranfer là cơ sở để thực hiện đánh giá giá trị của cầu thủ sau này

```
class Tranfer:
    def __init__(self,name,price,link) -> None:
        self.name=name
        self.price=price
        self.statLink=link

        self.skill=0
        self.pot=0

        self.matches=0
        self.minutes_play=0
        self.goals=0
        self.assists=0
        self.red_cards=0
        self.yellow_red_cards=0
        self.yellow_cards=0

        self.averageETV=0

    def setAttr(self,skill,pot,averageETV,matches,minutes_play,goals,assists,yellow_cards,yellow_red_cards,red_cards):
        self.skill=skill
        self.pot=pot
        self.averageETV=averageETV

        self.matches=matches
        self.minutes_play=minutes_play
        self.goals=goals
        self.assists=assists
        self.red_cards=red_cards
        self.yellow_red_cards=yellow_red_cards
        self.yellow_cards=yellow_cards
```

- Lớp có các thành phần cơ bản là tên cầu thủ(name), giá trị chuyển nhượng của cầu thủ(price)

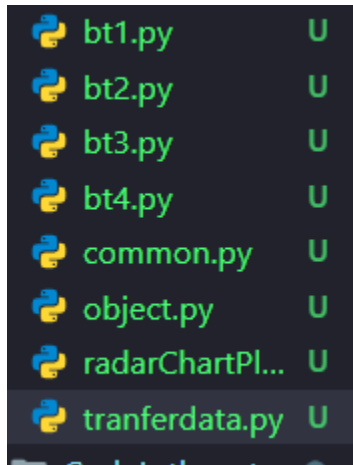
- Các giá trị còn lại là thành phần để dựa vào để định giá cầu thủ bao gồm:skill(kĩ năng), pot(tiềm năng),mathches(số ván chơi), minutes_play(số phút đã chơi),goal(số bàn thắng), assists(kiến tạo) và các loại thẻ đỏ vàng đã nhận, trong đó quan trọng nhất đó chính là averageETV là khoảng giá trị ước tính chuyển nhượng trung bình của một cầu thủ đóng vai trò quan trọng trong việc định giá cầu thủ

- Lớp còn có 1 hàm setAttr để thiết lập các thuộc tính cho cầu thủ

-Thuộc tính statLink chỉ là đường dẫn đến trang thông tin chi tiết về số liệu cầu thủ để sau này có thể lấy data về
















b) Thu thập thông tin cầu thủ

Ta tạo thêm 1 file mới tên là tranferdata.py chỉ để lấy thông tin từ trang web



Trong file này ta thực hiện như sau:

-Do trang web có thể dùng hậu tố như 'K','M',... để thể hiện giá trị chuyển nhượng

 Nicksoen Gomis	 Sheff Utd	▶  Toronto	23-02-2024	Free
 Richie Laryea D (R), M (CL)	 Nottingham	▶  Toronto	23-02-2024	€0.7M
 Bruno Jordão DM, M (C)	 Wolverhamp...	▶  Radomiak	06-02-2024	Free
 Charlie Wiggett D (C)	 Newcastle U...	▶  Sligo	02-02-2024	Free
 Serge Aurier D (R)	 Nottingham	▶  Galatasaray	02-02-2024	€0.1M

Ta cần 1 hàm để xử lý vấn đề này bằng cách biến đổi các kí hiệu đó thành 1 số cụ thể, hàm changePriceToNumber được ra đời để xử lý vấn đề này


```
def changePriceToNumber(price):
    if price == "Free":
        return 0
    if price[-1] == 'K':
        return int(float(price[1:-1]) * 1000)
    elif price[-1] == 'M':
        return int(float(price[1:-1]) * 1000000)
    return int(price[1:])
```

Hàm changePriceToNumber

-Do số lượng page khá lớn(18 trang) chứa thông tin về giá chuyển nhượng của các cầu thủ, ta sử dụng đa luồng để thực hiện lấy data một cách tối ưu nhất

```
idTable = "player-table-body"
list_result = []
threads = []
PAGE_QUANTITY = 18 # Tổng số trang
PAGES_PER_THREAD = 3 # Số trang mỗi luồng xử lý

# Tạo các luồng, mỗi luồng sẽ xử lý 3 trang
for i in range(0, PAGE_QUANTITY, PAGES_PER_THREAD):
    page_numbers = range(i + 1, min(i + PAGES_PER_THREAD + 1, PAGE_QUANTITY + 1))
    thread = threading.Thread(target=getTransferFromWeb, args=(page_numbers, idTable, list_result))
    threads.append(thread)
    thread.start()

# Chờ tất cả các luồng hoàn thành
for thread in threads:
    thread.join()

# list_result chứa kết quả từ tất cả các trang
print(f"Total Transfers: {len(list_result)}")
```

-Các trang chứa data đều nằm trong thẻ có id="player-table-body", nên idTable sẽ là biến truyền vào hàm getTransferFromWeb lấy data sau này sẽ giải thích chi tiết

-Số trang là 18, chạy mỗi luồng 3 trang như vậy sẽ có 6 luồng, kết quả được lưu vào mảng list_result chứa các thành phần là đối tượng Transfer

*Hàm getTransferFromWeb nhận 3 tham số là mảng các trang cần duyệt, id của bảng data và mảng kết quả

Dùng vòng lặp để chạy hết các trang có trong mảng các trang cần duyệt lấy được các tr_list chứa thông tin chuyển nhượng các cầu thủ có trong trang

```

def getTranferFromWeb(page_numbers, idTable, results):
    driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
    try:
        for page in page_numbers:
            url = f"https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league/transfers/2023-2024/{page}"
            driver.get(url)
            time.sleep(5)

            html_content = driver.page_source
            soup = BeautifulSoup(html_content, 'html.parser')

            # player
            tbody = soup.find('tbody', id=idTable)
            tr_list = tbody.find_all('tr')
            for tr in tr_list:
                tdName = tr.find('td')

                spanName = tdName.find('span')
                name = spanName.text
                price = tr.find_all('td')[-1].text

                #link
                link=''
                a=tdName.find('a', recursive=True)
                if a!=None:
                    link=a['href']

                results.append(Tranfer(name, changePriceToNumber(price),link))
            print(f"Finish Page {page}")
    finally:
        driver.quit()

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
try:
    for page in page_numbers:
        url = f"https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league/transfers/2023-2024/{page}"
        driver.get(url)
        time.sleep(5)

        html_content = driver.page_source
        soup = BeautifulSoup(html_content, 'html.parser')

        # player
        tbody = soup.find('tbody', id=idTable)
        tr_list = tbody.find_all('tr')

```

Duyệt từng phần tử trong tr_list, qua 1 vài bước xử lý, ta lấy được thông tin về tên, giá chuyển nhượng và đường dẫn đến trang chứa thông tin cầu thủ

-Sau khi có mảng kết quả, từ đường dẫn trong thuộc tính của đối tượng Tranfer, ta duyệt qua từng link để thiết lập các thuộc tính còn lại, việc này cũng được làm tương tự như trên, cũng chạy đa luồng với mỗi luồng chạy 85 trang thông qua hàm getStatPlayer nhận tham số là trang bắt đầu, trang kết thúc và mảng kết quả

-Cuối cùng lưu vào 1 file để sau này tiện làm việc mà không cần phải chạy lại những đường link trên để tránh mất thời gian cho mỗi lần khảo sát

```

NUMBEROFPLAYER=len(list_result)
PAGEPERTHREAD=120

threads = []
for i in range(0, NUMBEROFPLAYER, PAGEPERTHREAD):
    startIndex = i
    endIndex=min(i+PAGEPERTHREAD,NUMBEROFPLAYER)
    thread = threading.Thread(target=getStatPlayer, args=(startIndex, endIndex, list_result))
    threads.append(thread)
    thread.start()

# Chờ tất cả các luồng hoàn thành
for thread in threads:
    thread.join()

import pickle
file_path_tranfer=os.path.join(os.path.dirname(__file__), "trnfers.pkl")
with open(file_path_tranfer, "wb") as file:
    pickle.dump(list_result, file)

print("Get data success")

```

Thu thập dữ liệu và lưu vào file tranfers.pkl

Ngoài ra, ta còn thu thập thêm thông tin về độ tuổi của các cầu thủ, ở đây ta tận dụng api của web trả về thay vì truy cập vào từng link gây tốn thời gian.

Thu thập thông tin về độ tuổi và player_slug, player_slug giống như id của cầu thủ, mỗi cầu thủ có 1 slug riêng, slug này giống với đuôi cuối trong đường dẫn đến bảng thông tin chi tiết về cầu thủ(Thuộc tính statLink của đối tượng Tranfer)

Từng cầu thủ được lưu theo cặp tuổi, slug, cuối cùng lưu vào file **slugAndAge.pkl** phục vụ cho phần dự đoán cầu thủ sau

```

import json
import requests

form_data = {
    'orderBy': 'date_transfer',
    'orderByDescending': 1,
    'page': 1,
    'pages': 0,
    'pageItems': 500,
    'countryId': 'all',
    'season': 5847,
    'tournamentId': 31,
    'clubFromId': 'all',
    'clubToId': 'all',
    'transferFeeFrom': None,
    'transferFeeTo': None,
}

url = 'https://www.footballtransfers.com/en/transfers/actions/confirmed/overview'
res = requests.post(url, data=form_data)
file_slugAndAge = os.path.join(os.path.dirname(__file__), "slugAndAge.pkl")
if res.status_code == 200:
    json_data = res.json()
    import pickle
    name_slug_age=[]
    with open(file_slugAndAge, "wb") as file:
        for i in range(json_data['filter_records']):
            age=json_data['records'][i]['age']
            if age!=None and age.isdigit():
                age=int(age)
            else:age=0
            arr=[age,json_data['records'][i]['player_slug']]
            name_slug_age.append(arr)
        pickle.dump(name_slug_age,file)
else:
    print("Lỗi")

```

c) Đề xuất phương án định giá cầu thủ

-Sau khi đã có file dữ liệu từ phần trước, ta thực hiện import vào trong file bt4.py, thực hiện thiết lập giá trị cho các mảng thuộc tính

```

from object import Tranfer
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import pickle
import os

file_pathTranfer = os.path.join(os.path.dirname(__file__), "trangers.pkl")
file_slugAndAge = os.path.join(os.path.dirname(__file__), "slugAndAge.pkl")

list_tranfer=[]
name_slug_age=[]
with open(file_pathTranfer, "rb") as file:
    list_tranfer = pickle.load(file)
print(len(list_tranfer))
with open(file_slugAndAge, "rb") as file:
    name_slug_age = pickle.load(file)
print(len(list_tranfer))

```

Chuẩn bị các vector là thuộc tính của cầu thủ

```

for i in list_tranfer:
    price_arr.append(i.price)
    skill_arr.append(i.skill)
    pot_arr.append(i.pot)
    matches_arr.append(i.matches)
    minutesPlay_arr.append(i.minutes_play)
    goals_arr.append(i.goals)
    assists_arr.append(i.assists)
    yellowCards_arr.append(i.yellow_cards)
    yellowRedCards_arr.append(i.yellow_red_cards)
    redCards_arr.append(i.red_cards)
    averageETV_arr.append(i.averageETV)
    lastTranfer_arr.append(i.lastTranfer)
    age=0 #Xác định tuổi
    for j in name_slug_age:
        if i.statLink.split(r'/' )[-1]==j[1]:
            age=j[0]
    age_arr.append(age)

```

Đồng thời ta cũng lưu giá trị các thuộc tính của từng cầu thủ vào trong mảng player để tiện cho việc test sau này

```
player=[]
for i in list_tranfer:
    arr=[]
    arr.append(i.name)
    arr.append(i.price)
    arr.append(i.skill)
    arr.append(i.pot)
    arr.append(i.matches)
    arr.append(i.minutes_play)
    arr.append(i.goals)
    arr.append(i.assists)
    arr.append(i.yellow_cards)
    arr.append(i.yellow_red_cards)
    arr.append(i.red_cards)
    arr.append(i.averageETV)
    arr.append(i.lastTranfer)
    age=0 #Xác nhận tuổi
    for j in name_slug_age:
        if i.statLink.split(r'/')[1]==j[1]:
            age=j[0]
    arr.append(age)
    player.append(arr)
```

*Thực hiện hồi quy tuyến tính để đề ra phương pháp định giá cầu thủ

```

y = np.array([price_arr]).T
X1 = np.array([skill_arr]).T
X2 = np.array([pot_arr]).T
X3 = np.array([matches_arr]).T
X4 = np.array([minutesPlay_arr]).T
X5 = np.array([goals_arr]).T
X6 = np.array([assists_arr]).T
X7 = np.array([yellowCards_arr]).T
X8 = np.array([yellowRedCards_arr]).T
X9 = np.array([redCards_arr]).T
X10 = np.array([averageETV_arr]).T
X11 = np.array([lastTransfer_arr]).T
X12 = np.array([age_arr]).T

# Tạo ma trận đặc trưng mở rộng Xbar với cột 1
one = np.ones((X1.shape[0], 1)) # Cột 1 để tính phần bù
Xbar = np.concatenate((one,X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12), axis=1)

# Sử dụng Scikit-learn để huấn luyện mô hình hồi quy tuyến tính
regr = LinearRegression(fit_intercept=False)
regr.fit(Xbar, y)

print( 'Solution found by scikit-learn      : ', regr.coef_ )

```

Mảng y chính là mảng giá trị cầu thủ, Xbar là ma trận tạo bởi các giá trị thuộc tính của cầu thủ, qua quá trình huấn luyện ta thu được kết quả cuối cùng. Sau đây là demo 1 số kết quả

```

Solution found by scikit-learn      : [[-3.16389472e+06 -7.79505710e+04  2.14220829e+04  4.58192637e+03
-1.46451150e+02 -1.27495299e+04  1.85734101e+04  1.09738114e+04
-1.49252417e+05  8.60317260e+04  4.38388268e-02  9.89822240e-01
 2.80783893e+05]]
Nhập id cầu thủ cần kiểm tra: 196
Dự đoán giá cầu thủ Arthur Masuaku: 2367347 || Giá trị thực:  2000000
Độ lệch 367347

```

```

Solution found by scikit-learn      : [[-3.16389472e+06 -7.79505710e+04  2.14220829e+04  4.58192637e+03
-1.46451150e+02 -1.27495299e+04  1.85734101e+04  1.09738114e+04
-1.49252417e+05  8.60317260e+04  4.38388268e-02  9.89822240e-01
 2.80783893e+05]]
Nhập id cầu thủ cần kiểm tra: 122
Dự đoán giá cầu thủ Gustavo Hamer: 17186463 || Giá trị thực:  17300000
Độ lệch 113537

```

Có thể thấy có 1 số dữ liệu mô hình dự đoán gần đúng nhưng có một số thì độ lệch rất lớn hệ số xác định R^2 chỉ đạt giá trị cực hạn là 0.9431488656941758 nên mô hình có vẻ như dự đoán có sự sai khác khá lớn nhưng tổng quan thì vẫn có thể chấp nhận được

Ta có thể thực hiện kiểm tra xem độ lệch của mô hình như thế nào thông qua số lượng lệch ở từng mức

```
mini_different=0
small_different=0 #Độ lệch nhỏ hơn 2tr
mid_different=0 #Độ lệch nhỏ hơn 4tr lớn hơn 2tr
big_different=0 #Độ lệch nhỏ hơn 8tr lớn hơn 4tr
supper_different=0 #Độ lệch lớn hơn 8tr
for i in range(len(player)):
    new_player = np.array([[1]+player[i][2:]])
    predicted_price = regr.predict(new_player)
    tmp=abs(predicted_price[0,0]-player[i][1])
    if tmp<1000000: mini_different+=1
    elif tmp<2000000: small_different+=1
    elif tmp<4000000: mid_different+=1
    elif tmp<8000000: big_different+=1
    else: supper_different+=1

print("Độ lệch nhỏ hơn 1tr:",mini_different)
print("Độ lệch nhỏ hơn 2tr lớn hơn 1tr:",small_different)
print("Độ lệch nhỏ hơn 4tr lớn hơn 2tr:",mid_different)
print("Độ lệch nhỏ hơn 8tr lớn hơn 4tr:",big_different)
print("Độ lệch lớn hơn 8tr:",supper_different)
```

```
Độ lệch nhỏ hơn 1tr: 239
Độ lệch nhỏ hơn 2tr lớn hơn 1tr: 76
Độ lệch nhỏ hơn 4tr lớn hơn 2tr: 74
Độ lệch nhỏ hơn 8tr lớn hơn 4tr: 17
Độ lệch lớn hơn 8tr: 21
```

Qua số liệu ta có thể đánh giá độ tin cậy cho phương pháp này