HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

----------o0o----------

BÁO CÁO BÀI TẬP LỚN
PYTHON

Giảng viên: Kim Ngọc Bách

Nhóm môn học: 11

Sinh viên: Nguyễn Gia Hiệp

Mã sinh viên: B22DCCN297

Lớp: D22CQCN09_B

HÀ NỘI - 2024

MỤC LỤC

I. Viết chương trình Python thu thập dữ liệu phân tích cụ thể (file ‘btlon1.py’)

- Để có thể thu thập dữ liệu phân tích em sẽ chia code thành 4 phần:

+ Lấy các link chứa các thuộc tính

+ Đưa các thuộc tính vào 1 list

+ Truy cập lần lượt từng link và lấy các dữ liệu thuộc tính

+ Lọc và sắp xếp dữ liệu sau đó ghi ra file ‘results.csv’

1. Lấy các link chứa các thuộc tính

```
r=requests.get('https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats')
so=bs(r.content, 'html.parser')
p_thuoc_tinh=so.find_all('ul',class_='hoversmooth')
ul_thuoc_tinh=p_thuoc_tinh[1].find('ul',class_='')
link_thuoc_tinh=ul_thuoc_tinh.find_all('a')
list_cau_thu=[]
```

- Dòng đầu gửi một yêu cầu HTTP GET đến URL được chỉ định (https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats) bằng thư viện requests và lưu trữ phản hồi trong biến r

- so là 1 BeautifulSoup dùng để phân tích nội dung của phản hồi của r với trình cú pháp html parser

- Các dòng tiếp theo dùng để tìm kiếm đến đến các thẻ và class chứa các dữ liệu các link

- Dòng cuối cùng có tác dụng tạo 1 list để lưu các dữ liệu của cầu thủ

2. Đưa các thuộc tính vào 1 list

- Để thuận tiện cho việc lưu dữ liệu vào list với nhiều url thì em sẽ tạo 1 hàm để lưu với tham số đầu vào là 1 mã url và 1 chỉ số bắt đầu lấy dữ liệu

```
def luu(url,begin):
    time.sleep(10)
    resp=requests.get(url)
    soup=bs(resp.content,'html.parser')
    comments = soup.find_all(string=lambda text: isinstance(text, cm))
    len_ct=0
    check=False
    tmp=0
    if len(list_cau_thu)>0: tmp=len(list_cau_thu[0])
    for comment in comments:
        comment_soup = bs(comment, "html.parser")
        div=comment_soup.find('div',class_='table_container')
        table=comment_soup.find('table',class_='min_width sortable stats_table min_width shade_zero')
        if table!=None:
            tbody=table.find('tbody')
            trs=tbody.find_all('tr')
            for tr in trs:
                cau_thu=[]
                tds=tr.find_all('td')
                for td in tds:
                    a=td.find('a')
                    s=''
                    if a==None:   s=td.text
                    else:   s=a.text
                    cau_thu.append(s.replace(',',''))
                if len(cau_thu)!=0:
                    del cau_thu[-1]
                    if len(list_cau_thu)<580:list_cau_thu.append(cau_thu[begin:])
                    else:
                        check=True
                        len_ct=len(cau_thu)
                        for i in range(len(list_cau_thu)):
                            if list_cau_thu[i][0]==cau_thu[0] and list_cau_thu[i][3]==cau_thu[3] and list_cau_thu[i][4]==cau_thu[4]:
                                for j in range(begin,len(cau_thu)):
                                    if cau_thu[j]=='': list_cau_thu[i].append('N/a')
                                    else: list_cau_thu[i].append(cau_thu[j])
    for i in range(580):
        if tmp==len(list_cau_thu[i]):
            for j in range(begin,len_ct):
                list_cau_thu[i].append('N/a')
```

- Dòng đầu tiên có tác dụng delay 10 giây để phòng trường hợp lỗi 429

- 2 dòng tiếp theo có tác dụng gửi một yêu cầu HTTP GET đến url và phân tích nội dung của phản hồi với trình cú pháp HTML

- tmp dùng để lưu kích thước của cầu thủ trước khi thêm các dữ liệu chỉ số mới vào (len_ct và check là 2 biến dùng để debug nên trong code không có tác dụng gì)

- 580 là số cầu thủ tham gia giải bóng đá ngoại hạng Anh mùa 2023-2024

- Trong nguồn trang, bảng chỉ số của các cầu thủ nằm trong phần comment vì thế nên comments được tạo ra để đọc các đoạn code html nằm trong comment

- Duyệt lần lượt từng comment sau đó gọi 1 biến comment_soup để phân tích nội dung trong comment với trình phân tích cú pháp HTML

- div dùng lưu nội dung của thẻ div chứa bảng có dữ liệu chỉ số các cầu thủ

- table dùng lưu nội dung của bảng có dữ liệu chỉ số các cầu thủ

- Vì comments là 1 list có nhiều phần tử nên em sẽ xét xem table có rỗng không. Nếu rỗng có nghĩa là phần comment đấy không có bảng cần tìm, còn nếu không rỗng có nghĩa là bảng cần tìm đang nằm trong phần comment đấy và tiếp tục chạy chương trình

- Sau đấy lần lượt duyệt đến từng hàng và mỗi hàng duyệt đến từng cột để lấy dữ liệu các cầu thủ. Nhưng trước khi duyệt đến từng cột em sẽ tạo 1 list có tên là cau_thu để khi duyệt từng cột của hàng để lấy dữ liệu sau đó lưu dữ liệu vào list cầu thủ

- Khi đã duyệt xong em sẽ kiểm tra xem list cau_thu có rỗng không. Nếu không rỗng có nghĩa là hàng đấy là 1 cầu thủ, còn nếu không thì không phải là 1 cầu thủ

- Xóa dữ liệu cuối cùng của list cau_thu vì dữ liệu đấy không phải dữ liệu cần tìm

- Kiểm tra xem list_cau_thu đã có size là 580(đây là số lượng cầu thủ thi đấu ở NHA mùa 2023-2024) chưa:

+ Nếu chưa thỏa mãn thì em sẽ thêm cau_thu vào list_cau_thu
+ Nếu đã thỏa mãn thì em sẽ duyệt lần lượt các cầu thủ có trong list cầu thủ nếu phần tử đầu tiên(tên cầu thủ), phần tử thứ 4(CLB thi đấu), phần tử thứ 5(tuổi cầu thủ) của cau_thu ứng với 1 cầu thủ nào đó trong list_cau_thu thì ta sẽ thêm lần lượt các phần tử mới vào trong cầu thủ trong list_cau_thu với điều kiện nếu phần tử ấy không bằng ‘’ còn nếu bằng thì ta thêm phần tử ‘N/a’

- Cuối cùng ta kiểm tra xem trong list_cau_thu có những cầu thủ nào chưa được thêm dữ liệu mới vào thì ta thêm các dữ liệu là ‘N/a’ vào

3. Truy cập lần lượt từng link và lấy các dữ liệu thuộc tính

```
standard_stats='https://fbref.com'+link_thuoc_tinh[0].get('href')
luu(standard_stats,0)
print('1')
goalkeeping='https://fbref.com'+link_thuoc_tinh[1].get('href')
luu(goalkeeping,10)
print('2')
shooting='https://fbref.com'+link_thuoc_tinh[3].get('href')
luu(shooting,7)
print('3')
passing='https://fbref.com'+link_thuoc_tinh[4].get('href')
luu(passing,7)
print('4')
pass_type='https://fbref.com'+link_thuoc_tinh[5].get('href')
luu(pass_type,8)
print('5')
goal_shot='https://fbref.com'+link_thuoc_tinh[6].get('href')
luu(goal_shot,7)
print('6')
defensive='https://fbref.com'+link_thuoc_tinh[7].get('href')
luu(defensive,7)
print('7')
prosession='https://fbref.com'+link_thuoc_tinh[8].get('href')
luu(prosession,7)
print('8')
playing_time='https://fbref.com'+link_thuoc_tinh[9].get('href')
luu(playing_time,11)
print('9')
miscellaneous='https://fbref.com'+link_thuoc_tinh[10].get('href')
luu(miscellaneous,7)
print('10')
```

- Ta lần lượt tạo các url chứa các chỉ số cần tìm và gọi hàm luu để lưu các chỉ số cần tìm và truyền vào url và biến begin thích hợp với mỗi url (hàm print() em dùng để debug xem có url nào không chạy hay không nên đối với đoạn code nó sẽ không có tác dụng)

4. Lọc và sắp xếp dữ liệu sau đó ghi ra file ‘results.csv’

```
for i in range(len(list_cau_thu)-1,0,-1):
    if int(list_cau_thu[i][8].replace(',',''))<=90:
        list_cau_thu.remove(list_cau_thu[i])
```

- Câu lệnh trên có tác dụng để xóa những cầu thủ có số phút thi đấu ít hơn hoặc bằng 90 phút

```
s='Name,Nation,Position,Team,Age,matches played,starts,minutes,Assists,non-Penalty Goals,Penalty Goals,Yellow Cards,Red Cards,xG,npxG
arr=s.split(',')
```

- Gọi và lưu các chỉ số cần tìm trong đề bài (dùng để làm columns trong file ‘results.csv’)

```
for i in range(len(list_cau_thu)):
    list_cau_thu[i].pop(185)
    list_cau_thu[i].pop(184)
    list_cau_thu[i].pop(183)
    list_cau_thu[i].pop(182)
    list_cau_thu[i].pop(177)
    list_cau_thu[i].pop(176)
    list_cau_thu[i].pop(175)
    list_cau_thu[i].pop(174)
    list_cau_thu[i].pop(173)
    list_cau_thu[i].pop(172)
    list_cau_thu[i].pop(169)
    list_cau_thu[i].pop(168)
    list_cau_thu[i].pop(167)
    list_cau_thu[i].pop(21)
    list_cau_thu[i].pop(15)
    list_cau_thu[i].pop(12)
    list_cau_thu[i].pop(10)
    list_cau_thu[i].pop(9)
    list_cau_thu[i].pop(5)
```

- Xóa các dữ liệu không cần tìm trong đề bài

```
list_cau_thu.sort(key=lambda x:(x[0],-int(x[4])))
```

- Sắp xếp các cầu thủ theo tên theo thứ tự từ điển, nếu trùng tên sắp xếp theo số tuổi giảm dần

```
dataFrame=pd.DataFrame(list_cau_thu,columns=arr)
dataFrame.to_csv('results.csv')
```

- dataFrame là 1 DataFrame được tạo ra với các hàng là các phần tử trong list_cau_thu và tiêu đề cột là các phần tử trong arr sau đó ghi dataFrame vào file ‘results.csv’

II.

1. Tìm top 3 cầu thủ cao nhất và thấp nhất ở mỗi chỉ số (file ‘btlon2-1.py’)

- Để tìm top 3 cầu thủ cao nhất ở mỗi chỉ số em sẽ chia code thành 3 phần:

+ Đọc và chuyển hóa file ‘results.csv’ và các list

+ Biến đổi các phần tử ‘N/a’ trong list chứa dữ liệu của file ‘results.csv’ thành về kiểu float

+ Tìm top 3 cầu thủ cao nhất và thấp nhất ở mỗi chỉ số và ghi vào file ‘top_3_cau_thu.txt’

a, Đọc và chuyển hóa file ‘results.csv’ và các list

```
data=pd.read_csv('results.csv')
header=list(data.columns)
list_hang=data.values.tolist()
```

- data là 1 DataFrame dùng để lấy tất cả dữ liệu trong file ‘results.csv’, header là 1 list dùng để lưu các các tiêu đề của cột và list_hang dùng để lưu các dữ liệu của file

b, Biến đổi các phần tử ‘N/a’ trong list chứa dữ liệu của file ‘results.csv’ thành về kiểu float

```
for i in range(len(list_hang)):
    for j in range(len(list_hang[i])):
        if list_hang[i][j]=='N/a': list_hang[i][j]='0'
```

c, Tìm top 3 cầu thủ cao nhất và thấp nhất ở mỗi chỉ số và ghi vào file ‘top_3_cau_thu.txt’

- Vì trong trong list_hang sẽ có những phần tử mà không cần phải lấy top nên ta sẽ loại bỏ các phần tử đấy bằng cách duyệt từ phần tử thứ 6

```
for j in range(len(list_hang)):
    list_hang[j][i]=float(list_hang[j][i])
```

- 2 dòng trên có tác dụng biến đổi các phần tử cần sắp xếp về dạng float

```
list_hang.sort(key=lambda x:x[i])
```

- Sắp xếp các cầu thủ theo mỗi chỉ số

```
with open("top_3_cau_thu.txt", mode="a", encoding="utf-8") as file:
    file.write('min_'+header[i]+': ')
```

- Ghi vào file ‘top_3_cau_thu.txt’ với kí tự utf-8 và mode ‘a’ có thể ghi tiếp mà không cần phải xóa toàn bộ file và ghi lại

```
s_min=''
dem_min=0
for j in list_hang:
    if float(j[i])!=0:
        dem_min+=1
        s_min+=j[1]+'('+str(j[i])+')'+', '
    if dem_min==3:break
s_min=s_min[:-2]
with open("top_3_cau_thu.txt", mode="a", encoding="utf-8") as file:
    file.write(s_min+'\n')
    file.write('max_'+header[i]+': ')
```

- s_min là 1 chuỗi để lưu các cầu thủ nhỏ nhất, dem_min dùng để đếm số lượng cầu thủ

- Sau đó, duyệt các cầu thủ trong list_hang. Vì nếu có phần tử bằng 0 thì có thể đó là phần tử 'N/a' nên ta loại trường hợp đấy. Nếu phần tử không bằng 0 thì ta sẽ tăng biến dem_min thêm 1 và s_min thêm tên của cầu thủ và chỉ số tương ứng

- Nếu biến dem_min=3 thì em sẽ dừng vòng lặp

- Sau vòng lặp, em sẽ loại bỏ các kí tự dư thừa trong s_min

- Cuối cùng, em thêm các cầu thủ top 3 chỉ số nhỏ nhất vào trong file 'top_3_cau_thu.txt'

```
s_max=''
dem_max=0
for j in range(len(list_hang)-1,0,-1):
    if float(list_hang[j][i])!=0:
        dem_max+=1
        s_max+=list_hang[j][1]+'('+str(list_hang[j][i])+')'+', '
    if dem_max==3:break
s_max=s_max[:-2]
with open("top_3_cau_thu.txt", mode="a", encoding="utf-8") as file:
    file.write(s_max+'\n')
```

- Tương tự với cầu thủ có top 3 chỉ số lớn nhất

2. Tìm trung vị ở 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 (file 'btlon2-2.py')

- Trong phần này e sẽ chia code thành 4 phần:

+ Lọc và biến đổi các phần tử 'N/a' thành 0

+ Tìm trung vị, trung bình và độ lệch chuẩn của all

+ Tìm trung vị, trung bình và độ lệch chuẩn cho từng CLB

+ Ghi dữ liệu vào file ‘results2.csv’

a, Lọc và biến đổi các phần tử ‘N/a’ thành 0

```
data=pd.read_csv('results.csv')
data.replace('N/a',0,inplace=True)
numeric_df = data.select_dtypes(include=[float, int])
```

- data là DataFrame được đọc từ file ‘results.csv’

- Thay thế các phần tử ‘N/a’ thành 0 trong data

- numeric_df là DataFrame được tạo thành từ data bằng cách lọc các cột có dữ liệu là float và int trong data

b, Tìm trung vị, trung bình và độ lệch chuẩn của all

```
overall_stats = {
    f"{stat}_mean": numeric_df[stat].mean() for stat in numeric_df.columns
}
overall_stats.update({
    f"{stat}_median": numeric_df[stat].median() for stat in numeric_df.columns
})
overall_stats.update({
    f"{stat}_std": numeric_df[stat].std() for stat in numeric_df.columns
})
overall_stats['Team'] = 'all'
overall_stats_df = pd.DataFrame([overall_stats])
```

- overall_stats là 1 dictionaries.

- Ở 2 dòng đầu tiên, em tính trung bình các chỉ số rồi thêm vào overall_stats. Tiếp theo em lần lượt tính trung vị và độ lệch chuẩn các chỉ số rồi thêm vào overall_stats

- Sau đó em thêm key= ‘Team’ và gán giá trị là all để xác định trung bình, trung vị, độ lệch chuẩn của all

- overall_stats_df là 1 DataFrame được biến đổi từ overall_stats

c, Tìm trung vị, trung bình và độ lệch chuẩn cho từng CLB

```
team_mean = data.groupby('Team')[numeric_df.columns].mean().rename(columns=lambda x: x + '_mean')
team_median = data.groupby('Team')[numeric_df.columns].median().rename(columns=lambda x: x + '_median')
team_std = data.groupby('Team')[numeric_df.columns].std().rename(columns=lambda x: x + '_std')
team_stats = pd.concat([team_mean, team_median, team_std], axis=1).reset_index()
```

- team_mean là 1 DataFrame được tạo ra bằng cách phân nhóm theo cột ‘Team’ bằng hàm ‘groupby()’ của DataFrame data, sau đó chỉ định các cột trong numeric_df để tính toán. Hàm mean() dùng để tính trung bình các chỉ số của cột được nhóm và đặt lại tên cột bằng hàm rename()

- 2 dòng tiếp theo tương tự với median() là tính trung vị và std() là tính độ lệch chuẩn

- team_stats là 1 DataFrame được tạo ra bằng cách kết hợp team_mean, team_meadia, team_std lại với nhau theo cột bằng hàm concat và tham số axis=1. Sau đó dùng hàm reset_index() để thiết lập chỉ số

```
final_stats = pd.concat([overall_stats_df, team_stats], ignore_index=True)
final_stats = final_stats[['Team'] + [col for col in final_stats.columns if col != 'Team']]
final_stats=final_stats.drop(columns=['Unnamed: 0_mean','Unnamed: 0_std','Unnamed: 0_median'])
final_stats.to_csv('results2.csv', index=False)
```

- final_stats là 1 DataFrame được kết hợp từ over_stats_df và team_stats và được reset lại chỉ số bằng ignore_index=True

- Tiếp theo sắp xếp lại thứ tự cột theo cách đưa cột 'Team' là cột đầu tiên và các cột còn lại theo thứ tự như cũ

- Tiếp theo xóa các cột không cần thiết và lưu vào file 'results2.csv'

3. Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và toàn đội (file 'histogram.py')

- Trong bài này em sẽ chia code thành 2 phần:

+ Đọc dữ liệu từ file 'results2.csv'

+ Vẽ biểu đồ historgram từng chỉ số

a, Đọc dữ liệu từ file 'results2.csv'

```
data=pd.read_csv("results2.csv")
header=list(data.columns)
```

- data là 1 DataFrame để đọc dữ liệu từ file 'results2.csv', header là 1 list dùng để lưu trữ tiêu đề các cột trong file

b, Vẽ biểu đồ historgram từng chỉ số

```
for i in range(1,len(header)):
    plt.hist(data[header[i]],bins=10,color='skyblue',edgecolor='black')
    plt.title('Bảng '+ header[i])
    plt.xlabel(header[i])
    plt.ylabel('Mức độ')
    plt.show()
```

- Đầu tiên em duyệt từng chỉ số cần để vẽ biểu đồ historgram

- Với mỗi chỉ số, em dùng hàm plt.hist() để vẽ biểu đồ từng cột với số lượng cột là 10, màu của các cột là skyblue và màu của viền cột là black

- Hàm plt.title() để đặt tên cột, plt.xlabel() để đặt nhãn trục x, plt.ylabel() để đặt nhãn trục y, plt.show() để show biểu đồ

4. Tìm đội bóng có chỉ số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024 (file 'btlon2-3.py')

- Trong phần này em sẽ chia làm 2 phần:

+ Đọc file 'results2.csv' và khởi tạo các biến cần dùng để tìm đội bóng có điểm số cao nhất ở mỗi chỉ số

+ Tìm đội bòng có điểm số cao nhất ở mỗi chỉ số

a, Đọc file 'results2.csv' và khởi tạo các biến cần dùng để tìm đội bóng có điểm số cao nhất ở mỗi chỉ số

```
data=pd.read_csv('results2.csv')
header=list(data.columns)
ds=data.values.tolist()
dict={}
for i in ds:
    dict[i[0]]=0
file=open('chi_so_lon_nhat.txt','w')
```

- data là 1 DataFrame đùng để đọc file 'results.csv', header dùng để lưu tiêu đề của các cột trong data, ds dùng để lưu các dữ liệu trong data

- dict là 1 dict được khởi tạo ban đầu gồm key là các tiêu đề của cột data và value bằng 0, file dùng để viết vào trong file 'chi_so_lon_nhat.txt'

b, Tìm đội bòng có điểm số cao nhất ở mỗi chỉ số

```
for i in range(1,141):
    Max=0
    s=''
    for j in ds:
        if j[i]>Max:
            Max=j[i]
            s=j[0]
    file.write(header[i]+':'+s+'('+str(Max)+')\n')
    if i>1:dict[s]+=1
Max=max(dict.values())
file.write('\n\nDoi phong do tot nhat la: ')
for i in dict:
    if Max==dict[i]:
        file.write(i+', ')
```

- Em lần lượt duyệt các chỉ số cần tìm, với mỗi chỉ số em sẽ tìm giá trị lớn nhất của từng chỉ số bằng cách:

+ Duyệt lần lượt từng hàng tương ứng với mỗi đội bóng và all trong data

+ Tạo một biến Max để kiểm tra và gán giá trị lớn nhất vào biến Max và đội bóng tương ứng với giá trị đấy bằng biến s

+ Khi đã duyệt xong các đội bóng thì em sẽ viết vào file chỉ số và đội bóng với giá trị lớn nhất tương ứng

+ Sau đó em tăng value của key đội bóng có giá trị lớn nhất thêm 1 (nếu chỉ số đấy không phải là tuổi

- Tiếp theo em tìm xem đội bóng nào có số giá trị lớn nhất nhiều nhất rồi in đội bóng đấy vào file

III.

1. Sử dụng thuật toán K-means để phân loại cầu thủ thành các nhóm có chỉ số giống nhau (file 'K_mean.py')

- Trong bài này em sẽ chia làm 3 phần:

+ Đọc và lấy dữ liệu trong file 'results.csv'

+ Tạo hàm để vẽ K_means

+ Xử lí dữ liệu và đưa dữ liệu vào hàm để vẽ K_means

a, Đọc và lấy dữ liệu trong file 'results.csv'

```
data=pd.read_csv('results.csv')
data=data.drop(columns=['Unnamed: 0','Name','Nation','Position','Team'])
data.replace('N/a',0,inplace=True)
data=data.select_dtypes(include=[float, int])
```

- data là 1 DataFrame dùng để đọc và lấy dữ liệu từ file 'results.csv'

- Sau đó xóa các cột không cần thiết và đổi các phần tử có giá trị 'N/a' thành 0 và lọc các cột có kiểu dữ liệu là float và int

b. Tạo hàm để vẽ K_means

```
def kmeans_display(X, label):
    X0 = X[label == 0, :]
    X1 = X[label == 1, :]
    X2 = X[label == 2, :]
    plt.plot(X0[:, 0], X0[:, 1], 'b^', markersize = 4, alpha = .8)
    plt.plot(X1[:, 0], X1[:, 1], 'go', markersize = 4, alpha = .8)
    plt.plot(X2[:, 0], X2[:, 1], 'rs', markersize = 4, alpha = .8)
    plt.axis('equal')
    plt.plot()
    plt.show()
```

- Hàm có các tham số đầu vào: X là tập dữ liệu đầu vào dưới dạng một mảng 2D với các điểm dữ liệu, label là một mảng chứa nhãn của mỗi điểm dữ liệu, cho biết điểm đó thuộc cụm nào sau khi phân cụm

- Tạo 1 mảng con X0 từ X chứa các điểm dữ liệu có nhãn là 0. Tương tự với X1 và X2 với nhãn là 1 và 2

- Tiếp theo, vẽ lần lượt các điểm X0, X1, X2 trên đồ thị lần lượt dùng hình tam giác xanh, hình tròn xanh lá cây và hình vuông màu đỏ để biểu diễn, kích thước các điểm là 4px, độ mờ các điểm là 0.8

- plt.axis() để đảm bảo trục x, y có cùng tỉ lệ, plt.plot() để vẽ đồ thị và plt.show() để hiển thị đồ thị

c, Xử lí dữ liệu và đưa dữ liệu vào hàm để vẽ K_means

```
means =data.values.tolist()
data_means=np.array(means)
cov=np.cov(data_means,rowvar=False)
kmeans = KMeans(n_clusters=3, random_state=0).fit(data_means)
pred_label = kmeans.predict(data_means)
kmeans_display(data_means, pred_label)
```

- means là 1 list được tạo ra từ các dữ liệu trong data

- data_means là 1 mảng NumPy được tạo ra từ means để dễ thuận tiện thao tác hơn

- cov là ma trận hiệp phương sai được tính từ data_means

- Tạo một mô hình K-Means với 3 cụm và huấn luyện mô hình trên dữ liệu data_means

- Dự đoán nhãn cụm cho từng điểm dữ liệu trong data_means dựa trên mô hình K-Means đã huấn luyện

- Gọi hàm để vẽ K_means

2. Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm?Vì sao?Bạn có nhận xét gì về kết quả (file 'btlon3-1.py')

- Trong bài này em sẽ chia làm 4 phần:

+ Đọc và xử lí dữ liệu

+ Khởi tạo các biến cần dùng

+ Chạy vòng lặp thực hiện K_means với số lượng cụm khác nhau

+ Vẽ biểu đồ phương pháp Elbow

a, Đọc và xử lí dữ liệu

```
data=pd.read_csv('results.csv')
data.replace('N/a',0,inplace=True)
data=data.drop(columns=['Unnamed: 0'])
df=data.select_dtypes(include=[float,int])
```

- data là 1 DataFrame dùng để đọc dữ liệu của file 'results.csv'

- Chuyển đổi dữ liệu và xóa các cột không cần thiết trong data

- Gọi 1 DataFrame có tên là df để lọc và lấy các cột có kiểu dữ liệu int và float có trong data

b, Khởi tạo các biến cần dùng

```
n_samples = len(df)
wcss = []
range_clusters = range(2, min(11, n_samples))
```

- n_samples dùng để lưu số lượng hàng trong df

- wcss là 1 list dùng để lưu trữ WCSS

c, Chạy vòng lặp thực hiện K_means với số lượng cụm khác nhau

```
for k in range_clusters:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df)
    wcss.append(kmeans.inertia_)
```

- Chạy 1 vòng lặp từ 2 đến min(11,n_samples) tương đương với số lượng cụm từ 2 đến min(11,n_samples), trong mỗi vòng lặp có:

+ Tạo mô hình K_means mới k cụm, 'random_state=42' đảm bảo tính lặp lại của kết quả, 'n_init=10' thuật toán K-Means chạy 10 lần với các centroids ban đầu khác nhau để chọn kết quả tốt nhất

+ Huấn luyện mô hình K_means trên dữ liệu df với hàm ‘kmeans.fit()’

+ Thêm giá trị ‘kmeans.inertia’ vào wcss

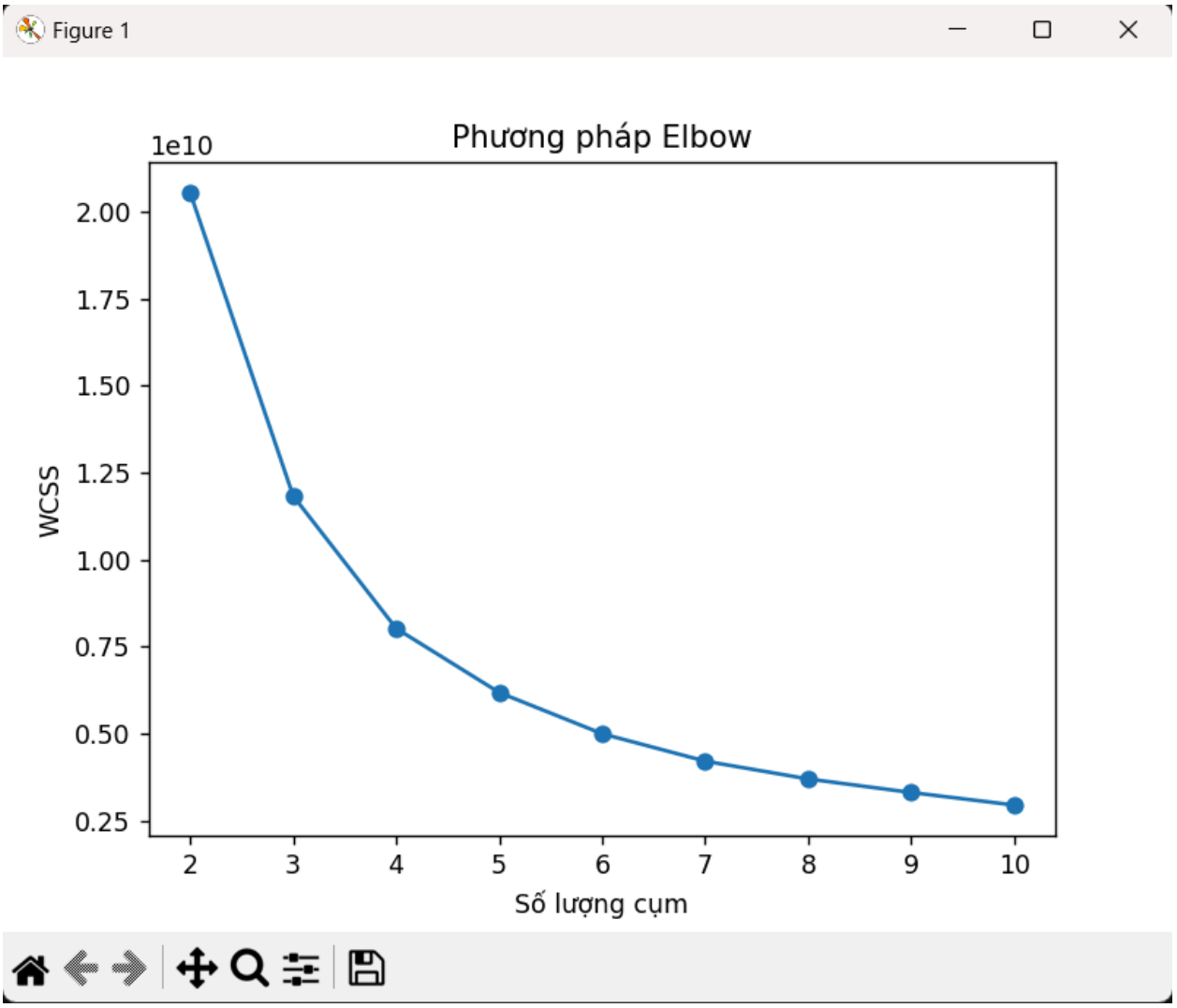
d, Vẽ biểu đồ phương pháp Elbow

```
plt.plot(range_clusters, wcss, marker='o')
plt.title('Phương pháp Elbow')
plt.xlabel('Số lượng cụm')
plt.ylabel('WCSS')
plt.show()
```

- Vẽ một đồ thị đường biểu diễn giá trị wcss (tổng bình phương trong cụm) theo số lượng cụm

- Thêm các tiêu đề đồ thị, nhãn trục x, nhãn trục y và show đồ thị

- Đồ thị sau khi chạy xong code:



- Dựa theo đồ thị thì ở điểm thứ 3 có WCSS giảm nhanh nhất, rồi bắt đầu chậm lại nên số cụm tối ưu mà em sẽ chọn là 3

3. Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D (file ‘pca_3.py’)

- Trong bài này em sẽ chia làm 5 phần:

+ Đọc và xử lí dữ liệu

+ Khởi tạo và huấn luyện mô hình K-Means

+ Giảm chiều dữ liệu bằng PCA

+ Phân cụm và dự đoán nhãn

+ Vẽ biểu đồ 2D

a, Đọc và xử lí dữ liệu

```
data = pd.read_csv('results.csv')
data=data.drop(columns=['Unnamed: 0','Name','Nation','Position','Team'])
data.replace('N/a',0,inplace=True)
```

- data là 1 DataFrame được tạo ra để đọc và lưu dữ liệu trong file ‘results.csv’

- Xóa các cột không cần thiết và thay đổi ‘N/a’ thành 0

b, Khởi tạo và huấn luyện mô hình K-Means

```
kmeans=KMeans(n_clusters=3,random_state=42)
```

- Tạo 1 mô hình K_means có số cụm là 3 và đảm bảo kết quả phân cụm ổn định qua các lần chạy

c, Giảm chiều dữ liệu bằng PCA

```
pca = PCA(n_components=2)
X_r = pca.fit_transform(data)
```

- Tạo 1 đối tượng pca để giảm chiều dữ liệu xuống là 2 chiều

- fit_transform(data): Huấn luyện mô hình PCA trên dữ liệu data và chuyển đổi dữ liệu sang không gian 2 chiều (X_r), cho phép vẽ đồ thị dễ dàng

d, Phân cụm và dự đoán nhãn

```
y=kmeans.fit_predict(data)
```

- fit_predict(data): Huấn luyện mô hình K-Means trên dữ liệu và dự đoán nhãn cụm cho mỗi điểm dữ liệu. y là một mảng chứa nhãn cụm tương ứng với từng điểm

e, Vẽ biểu đồ 2D

```
colors = ['navy','turquoise','darkorange']
plt.figure()
for i in range(3):
    plt.scatter(X_r[y==i, 0], X_r[y==i, 1],color=colors[i], alpha=0.8, lw=2)
plt.title('PCA plot of IRIS dataset')
plt.show()
```

- color là 1 list được lưu với các chuỗi kí tự tương ứng với các màu khác nhau

- plt.figure() dùng để tạo 1 khung vẽ đồ thị

- Chạy vòng lặp từ 0 đến 2 để vẽ các điểm dữ liệu lên đồ thị. Dữ liệu được chia theo cụm (y == i), với mỗi cụm được hiển thị bằng màu khác nhau.X_r[y == i, 0] và X_r[y == i, 1]: Tọa độ của các điểm dữ liệu trong không gian 2 chiều sau khi giảm chiều bằng PCA

- Đặt tiêu đề cho đồ thị bằng hàm plt.title() và show bằng plt.show()

4. Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ (file 'randa.py')

- Trong bài này em sẽ chia làm 5 phần:

+ Đọc và xử lí dữ liệu

+ Tạo hàm tìm vị trí cầu thủ theo tên (vì em so sánh các cầu thủ theo tên)

+ Chọn cầu thủ và chỉ số cần so sánh

+ Chuẩn bị dữ liệu để vẽ biểu đồ radar

+ Vẽ biểu đồ radar

a, Đọc và xử lí dữ liệu

```
data=pd.read_csv('results.csv')
data=data.drop(columns=['Unnamed: 0',  'Nation', 'Position', 'Team'])
data_list=data.values.tolist()
header=list(data.columns)
```

- Tạo DataFrame data để đọc dữ liệu từ file 'results.csv'

- Xóa các cột không cần thiết

- Tạo list data_list để lưu các giá trị trong data, header để lưu các tiêu đề cột trong data

b, Tạo hàm tìm vị trí cầu thủ theo tên

```
def vitri(name):
    for i in range(len(data_list)):
        if name==data_list[i][0]:   return i
    return -1
```

- Hàm sẽ trả về vị trí của cầu thủ nếu tồn tại tên trong data_list, nếu không tồn tại trả về '-1'

c, Chọn cầu thủ và chỉ số cần so sánh

```
i=-1
j=-1
while i==-1 or j==-1:
    print('Vui long nhap ten')
    i=vitri(input())
    j=vitri(input())
categories=header[9:16]
player1_stats=data_list[i][9:16]
player2_stats=data_list[j][9:16]
```

- Khởi tạo 2 biến i, j và khởi tạo giá trị ban đầu là -1 (có nghĩa là lúc đầu không có cầu thủ nào thỏa mãn

- Chạy vòng lặp đến khi nào tìm được tên cầu thủ thỏa mãn

- Tạo list categories để lưu các tiêu đề cột cần so sánh, player1-stats và player2-stats lần lượt lưu các giá trị cần so sánh

d, Chuẩn bị dữ liệu để vẽ biểu đồ radar

```
num_vars = len(categories)
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
player1_stats += player1_stats[:1]
player2_stats += player2_stats[:1]
angles += angles[:1]
```

- num_vars: Số lượng chỉ số cần vẽ

- angles: Mảng chứa góc tương ứng cho mỗi chỉ số trên biểu đồ radar

- Thêm phần tử đầu tiên vào cuối player1_stats, player2_stats, và angles để biểu đồ radar được khép kín

e, Vẽ biểu đồ radar

```
fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
ax.fill(angles, player1_stats, color='blue', alpha=0.25, label="Cầu thủ 1")
ax.fill(angles, player2_stats, color='red', alpha=0.25, label="Cầu thủ 2")
ax.plot(angles, player1_stats, color='blue', linewidth=2)
ax.plot(angles, player2_stats, color='red', linewidth=2)
ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories)
plt.title("So sánh chỉ số của hai cầu thủ")
ax.legend(loc='upper right', bbox_to_anchor=(1.1, 1.1))
plt.show()
```

- fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True)): Tạo một biểu đồ có trục tọa độ cực (radar plot)

- ax.fill(): Tô màu vùng biểu diễn chỉ số của cầu thủ 1 với màu xanh dương, và cầu thủ 2 với màu đỏ

- ax.plot(): Vẽ đường nối giữa các điểm chỉ số của cầu thủ 1 và cầu thủ 2

- ax.set_xticks(angles[:-1]): Thiết lập các vị trí góc để đặt nhãn

- ax.set_xticklabels(categories): Gắn nhãn các góc với tên chỉ số

- plt.title(): Đặt tiêu đề cho biểu đồ

- ax.legend(): Thêm chú thích để phân biệt các cầu thủ

- plt.show(): hiển thị biểu đồ

IV. Thu thập giá chuyển nhượng các cầu thủ trong mùa 2023-2024 của giải ngoại Hạng Anh