HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

-----ꕥ🕮ꕥ-----

BÁO CÁO BÀI TẬP LỚN

Lập trình Python

Sinh viên thực hiện:

Họ và tên : Lương Minh Quý

Mã SV: B22DCKH099

Hà Nội, ngày 3/11/2023

- Câu 1: Thu thập dữ liệu thống kê [*] của tất cả các cầu thủ có số phút thi đấu nhiều hơn 90 phút tại giải bóng đá ngoại hạng Anh mùa 2023-2024.
  - Ý tưởng :
    - Sử dụng thư viện BeautifulSoup và requests để lấy dữ liệu từ trên trang web về
    - Cách cài đặt : dung lệnh “pip install BeautifulSoup requests” để cài đặt hai thư viện
    - Ta chia những chỉ số thành nhiều phần trong cùng 1 table để dễ dàng sử lý
    - Ta có thể chia các chỉ số thành 10 phần sau:
      + Base_data
      + Goalkeep_data
      + Shooting_data
      + Passing_data
      + Goalshot_data
      + Defensive_data
      + Posess_data
      + Playtime_data
      + Miscell_data
  - Đầu tiên , ta thu thập dữ liệu về các đội bóng tham gia ngoại hạng Anh bằng 2 thue viện nói trên (HÌnh 1.1)

```
url='https://fbref.com/en/comps/9/2023-2024/2023-2024-Premier-League-Stats'
r=requests.get(url)
soup=BeautifulSoup(r.content,'html.parser')
table=soup.find('table',{
    'class': 'stats_table sortable min_width force_mobilize',
    'id': 'results2023-202491_overall'
})
#Danh sách chứa các đội bóng và url đến đội bóng đó
teams_data=[]
#Tìm các thẻ <a> trong <tbody> của table
tbody = table.find('tbody')
teams = tbody.find_all('a', href=True)
#Đưa dữ liệu về tên và link đội bóng vào danh sách
for team in teams :
    if "squads" in team['href']:
        team_name=team.text.strip()
        team_url="https://fbref.com" + team['href']
        teams_data.append([team_name,team_url])
```

(Hình 1.1)

  - Tiếp theo tạo 10 danh sách ứng vs 10 phần dữ liệu đã được chia ra (Hình 1.2)

```
# Tạo danh sách chứa dữ liệu của các hàng
base_data=[]
goalkeep_data=[]
shooting_data=[]
passing_data=[]
passtype_data=[]
goalshot_data=[]
defensive_data=[]
possess_data=[]
playtime_data=[]
miscell_data=[]
```

(Hình 1.2)

- Từ danh sách các đội tìm được , tiến hành lấy thông tin từ các đội ở từng bảng và đưa vào danh sách , mỗi data sẽ tương tứng vs 1 hàm riêng , minh họa hàm tìm base_data (HÌnh 1.3)

```
# Bắt đầu cào base_data
# Tìm bảng chứa dữ liệu
table=soup.find('table',{
    'class':'stats_table sortable min_width',
    'id':'stats_standard_9'
})
#Tìm tên các cột
if ok==1:
    b=[]
    b.append("Team")
    thead=table.find('thead').find_all('tr')
    for x in thead[1]:
        if x.text.strip()!="":
            b.append(x.text.strip())
    base_data.append(b)
#TÌm các dữ liệu còn lại của bảng
tbody=table.find('tbody').find_all('tr')
for y in tbody:
    tmp=[]
    tmp.append(team[0])
    for x in y:
        if(x.text.strip()==""):
            tmp.append("N/a")
        else :
            tmp.append(x.text.strip())
    base_data.append(tmp)
#Mấy cái dưới tương tự , thay đổi mỗi cái id bảng
#kết thúc cào base_data
```

(Hình 1.3)

- Sau khi tìm được tất cả các dữ liệu cần thiết , t chuyển tất cả các list tìm được thành dataframe và đưa hết vào 1 list chứa , minh họa cho vc chuyển list base_data thành df và đưa vào list chưa, những hàm khác tương tự (HÌnh 1.4)

```
#Tạo 1 list chứa tất cả các df
all_df=[]
#Chuyển đổi list thành Dataframe và loại bộ những hàng không cần thiết
base_data_df=pd.DataFrame(base_data[1:],columns=base_data[0])
base_data_df=base_data_df.drop(['90s','Gls','G+A','PK','npxG+xAG','Matches'],axis=1)
all_df.append(base_data_df.drop_duplicates(subset='Player'))
```

(Hình 1.4)

- Cuối cùng , ta lấy tổng hợp tất cả các df bằng outer join đòng thời loại bỏ trùng lặp và them điều kiện số phút thi đấu phải lớn hơn 90 (Hình 1.5)

```
#Outer join hết tất cả các df trong list all_df
result = reduce(lambda left, right: pd.merge(left, right, on=['Player'], how='outer'),all_df)
#Loại bỏ trùng lặp
result=result.drop_duplicates()
result = result.groupby('Player').first().reset_index()
result=result.T.drop_duplicates().T
#Loại các cầu thủ có chỉ số Min bé hơn 90 hoặc N/a
def ch(a:str):
    if a=='N/a' :
        return 0
    return int(a.replace(',',''))
a=list(result['Min'].values)
b=[x for x in a if ch(x)>90]
result=result[result['Min'].isin(b)].reset_index(drop=True)
print(result)
#Ghi thông tin vào file csv
# result.to_csv("results.csv",index=False)
```

(Hình 1.5)

- Câu 2 : Từ những thông tin từ file results.cvs , thực hiện các yêu cầu
- Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số (Hình 2.1)

  + Ý tưởng :

  _ Đầu tiên ta lọc ra các column có kiểu số

  _ TIếp đó tìm ra 3 cầu thủ có chỉ số cao nhất và thấp nhất ứng với tùng column và đưa vào list_column

  _tạo 1 dataframe được tạo nên từ các list column

```python
#Hàm tìm kiếm top 3 cầu thủ đối với từng chỉ số
def get_top_3(df):
    # #Tìm kiếm các cột kiểu số và đưa vào danh sách
    numeric_columns = df.select_dtypes(include=['number'])
    numeric_columns_list = numeric_columns.columns.tolist()
    def values(column:str):
        df_sorted = df.sort_values(by=column)
        a=list(df_sorted['Name'].head(3).values)
        b=list(df_sorted['Name'].tail(3).values)
        return (a+b)
    result_top=pd.DataFrame()
    for x in numeric_columns:
        result_top[x]=values(x)
    print("top3 và bottom 3 cầu thủ của từng chỉ số")
    print(result_top)
```

(Hình 2.1)

+ Kết quả sau khi chạy code :

```
top3 và bottom 3 cầu thủ của từng chỉ số
                Age                 MP              Starts                Goals              Assists1  ...                 OG               Recov                 Won                Lost                Won%
0       Mikey Moore        Jonny Castro        Zack Nelson  Aleksandar Mitrović           Alex Matos  ...     Đorđe Petrović       Andrey Santos     Aaron Ramsdale         Zack Nelson       Anass Zaroury
1     Ethan Nwaneri   Joshua Acheampong      Andrey Santos           Alex Iwobi           Alex Iwobi  ...       Aaron Hickey   Antwoine Hackford  Thomas Strakosha          Alex Iwobi     Andros Townsend
2      Leon Chiwome       Amadou Diallo  Antwoine Hackford           Alex Matos  Aleksandar Mitrović  ...     Aaron Ramsdale       Mason Burstow  Antwoine Hackford      Stefan Ortega   Antwoine Hackford
3  Yunus Emre Konak    Yunus Emre Konak   Yunus Emre Konak     Yunus Emre Konak     Yunus Emre Konak  ...   Yunus Emre Konak    Yunus Emre Konak   Yunus Emre Konak    Yunus Emre Konak         Zack Nelson
4        Zak Sturge          Zak Sturge         Zak Sturge           Zak Sturge           Zak Sturge  ...         Zak Sturge          Zak Sturge         Zak Sturge          Zak Sturge          Zak Sturge
5   Álvaro Carreras     Álvaro Carreras    Álvaro Carreras      Álvaro Carreras      Álvaro Carreras  ...    Álvaro Carreras     Álvaro Carreras    Álvaro Carreras     Álvaro Carreras     Álvaro Carreras

[6 rows x 154 columns]
```

- 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 :
  _ Đối với toàn bộ các cầu thủ thì ta có 3 bảng tìm về median,mean,std đối với mỗi chỉ số, sau khi tìm được thì t chỉ cần nối column của chúng lại tạo thành 1 dataframe1 mới (Hình 2.2)

```python
#Tìm trung vị,trung bình và độ lệch chuẩn mỗi chỉ số của toàn cầu thu
median_all = numeric_columns.median()
mean_all = numeric_columns.mean()
std_all = numeric_columns.std()
#ghộp tất cả thành 1 bảng
overall_df = pd.DataFrame({
    'STT': [0],
    'Team': ['all'],
    **{f'Median of {col}': [median_all[col]] for col in numeric_columns},
    **{f'Mean of {col}': [mean_all[col]] for col in numeric_columns},
    **{f'Std of {col}': [std_all[col]] for col in numeric_columns}
})
```

(Hình 2.2)

_ Còn đối vs đội cũng vậy , ta cũng có 3 bảng về median,mean,std đối vs mỗi chỉ số được group theo “team” , từ đó ra nối column chúng lại tạo thành 1 dataframe2 mới (Hình 2.3)

```
#Tìm trung vị , trung bình , độ lệch chuẩn của mỗi chỉ số theo đội
median_team = df.groupby('Team')[numeric_columns_list].median().round(2)
mean_team = df.groupby('Team')[numeric_columns_list].mean().round(2)
std_team = df.groupby('Team')[numeric_columns_list].std().round(2)
#Gộp các bảng này thành 1 bảng
team_df = pd.DataFrame({
    'STT': range(1, len(median_team) + 1),
    'Team': median_team.index,
    **{f'Median of {col}': median_team[col].values for col in numeric_columns},
    **{f'Mean of {col}': mean_team[col].values for col in numeric_columns},
    **{f'Std of {col}': std_team[col].values for col in numeric_columns}
})
```

(Hình 2.3)

_Cuối cùng ta nối df1 và df2 theo hang bằng hàm concat và them dữ liệu vào file result2.csv (Hình 2.4)

```
#Gộp hai bảng thành 1
final_df = pd.concat([overall_df, team_df], ignore_index=True)
final_df.to_csv("results2.csv")
```

(Hình 2.4)

- Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội (Hình 2.5)
  + Dầu tiên ta vẽ biểu đồ cho toàn giải (Hình 2.5.1) , sau đó ta tạo them thư mục để lưu trữ từng biểu đồ một (Hình 2.5.2)

```python
# Tên thư mục để lưu trữ các biểu đồ toàn giải
output_folder_1 = "histograms_all"

# Tạo thư mục nếu chưa tồn tại
if not os.path.exists(output_folder_1):
    os.makedirs(output_folder_1)

# Vẽ histogram cho toàn giải
for col in df:
    plt.figure(figsize=(8, 6))
    sns.histplot(df[col], bins=20, kde=True, color='blue')
    plt.title(f'Histogram of {col} - Toàn Giải')
    plt.xlabel(col)
    plt.ylabel('Số lượng cầu thủ (Người)')
    plt.grid(True, linestyle='--', alpha=0.5)
    # Lưu biểu đồ vào thư mục "histograms_all"
    plt.savefig(os.path.join(output_folder_1, f"{df.columns.get_loc(col)}.png"))
    plt.close()

print("Đã vẽ xong biểu đồ cho toàn giải")
```

(Hình 2.5.1)

```python
# Tên thư mục để lưu trữ các biểu đồ các đội
output_folder_2 = "histograms_teams"

# Tạo thư mục nếu chưa tồn tại
if not os.path.exists(output_folder_2):
    os.makedirs(output_folder_2)

# Vẽ histogram cho từng đội
teams = df['Team'].unique()
for team in teams:
    # Tên thư mục của đội
    team_folder = os.path.join(output_folder_2, team)
    # Tạo thư mục nếu chưa tồn tại
    if not os.path.exists(team_folder):
        os.makedirs(team_folder)

    team_data = df[df['Team'] == team]

    for col in df:
        plt.figure(figsize=(8, 6))
        sns.histplot(team_data[col], bins=20, kde=True, color='green')
        plt.title(f'Histogram of {col} - {team}')
        plt.xlabel(col)
        plt.ylabel('Số lượng cầu thủ (Người)')
        plt.grid(True, linestyle='--', alpha=0.5)
        # Lưu biểu đồ vào thư mục của đội
        plt.savefig(os.path.join(team_folder, f"{df.columns.get_loc(col)}.png"))
        plt.close()

    print(f"Đã vẽ xong biểu đồ cho đội {team}")
    time.sleep(3)

print("<<<<<<<<<Đã vẽ xong biểu đồ cho toàn giải và từng đội>>>>>>>>>>")
```

(HÌnh 2.5.2)

- Tìm các đội bóng có chỉ số cao nhất ở mỗi chỉ số
  + Ý tưởng t duyêt qua từng chỉ số của df1 (được tạo ra từ vc grouby ”team” với phương thức mean) và lưu lại vào trong list tên đội , chỉ số, giá trị max của chỉ số đó

```python
def get_best_team(df):
    #chuan bi
    results=[]
    numeric_columns = df.select_dtypes(include=['number'])
    numeric_columns_list = numeric_columns.columns.tolist()
    mean_team = df.groupby('Team')[numeric_columns_list].mean().round(2)
    for x in numeric_columns_list :
        team=mean_team[x].idxmax()
        value=mean_team[x].max()
        results.append([team,x,value])
    df_results=pd.DataFrame(results,columns=["Teams","Status","Value"])
    # print(df_results)
    # Đếm tần suất của từng đội
    team_counts = Counter([row[0] for row in results])
    # Chuyển kết quả đếm tần suất thành dạng bảng và sắp xếp nó
    frequency_table = [[team, count] for team, count in team_counts.items()]
    frequency_table.sort(key=lambda x: x[1], reverse=True)
    #In ra Teams có điểm số cao nhất
    print(frequency_table[0][0],frequency_table[0][1])
```

(Hình 2.6)

	Teams	Status	Value
0	Fulham	Age	27.16
1	Fulham	MP	23.52
2	Arsenal	Starts	16.72
3	Manchester City	Goals	3.76
4	Manchester City	Assists1	2.76
..	...	...	...
149	Sheffield Utd	OG	0.20
150	Everton	Recov	75.77
151	Everton	Won	26.00
152	Luton Town	Lost	22.54
153	Nott'ham Forest	Won%	56.16

(HÌnh 2.7 : bảng df_re sults)

- Từ hình 2.6 , ta thấy rằng việc tiếp theo là tìm tần xuất điểm của các đội bằng cách dung hàm couter để đếm tần xuất và sort để tìm giá trị lớn nhất

  + Kết quả của thu được sau khi chạy code : `Manchester City 61`

  ➔ Manchester City là đội có phong độ tốt nhất trong giải ngoại hạng Anh
- Trong hàm main , ta vt lại các hàm theo từng truy vấn (Hinh 2.8)

```
if __name__ == "__main__":
    df=pd.read_csv("results.csv")
    print("Chọn chức năng muốn thực hiện: ")
    print("1. Tìm Top 3 người có chỉ số cao nhất và thấp nhất")
    print("2. Tính trung vị, trung bình và độ lệch chuẩn của các chỉ số của toàn giải và các đội")
    print("3. Vẽ biểu đồ histogram cho toàn giải và từng đội")
    print("4. Tìm đội có giá trị cao nhất ở từng chỉ số và tần suất của từng đội và đánh giá")
    print("5. Thoát chương trình")
    while True:
        choice = int(input("Nhập lựa chọn của bạn: "))
        while choice < 1 or choice > 5:
            choice = int(input("Vui lòng nhập lại: "))
        if choice == 1:
            get_top_3(df)
        elif choice == 2:
            get_statistics(df)
        elif choice == 3:
            print_historgram(df)
        elif choice == 4:
            get_best_team(df)
        else:
            break
```

(Hình 2.8)

- Câu 3:

- Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau.
- Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.

  + Hàm vẽ biểu đồ (HÌnh 3.1)

```
def plot_kmeans(data, centroids, clusters, step):
        plt.figure(figsize=(8, 6))

        # Tạo màu sắc ngẫu nhiên cho các cụm
        colors = plt.cm.get_cmap('viridis', k)  # Sử dụng colormap 'viridis' với k màu

        # Vẽ các điểm dữ liệu theo cụm
        for i in range(k):
            points = data[clusters == i]
            plt.scatter(points[:, 0], points[:, 1], s=50, color=colors(i), label=f'Cluster {i}')
            plt.scatter(centroids[i, 0], centroids[i, 1], s=200, color=colors(i), marker='X', edgecolor='k')

        plt.title('K-means Clustering of Football Players')
        plt.xlabel('PC1')
        plt.ylabel('PC2')
        plt.legend()
        plt.show()
```

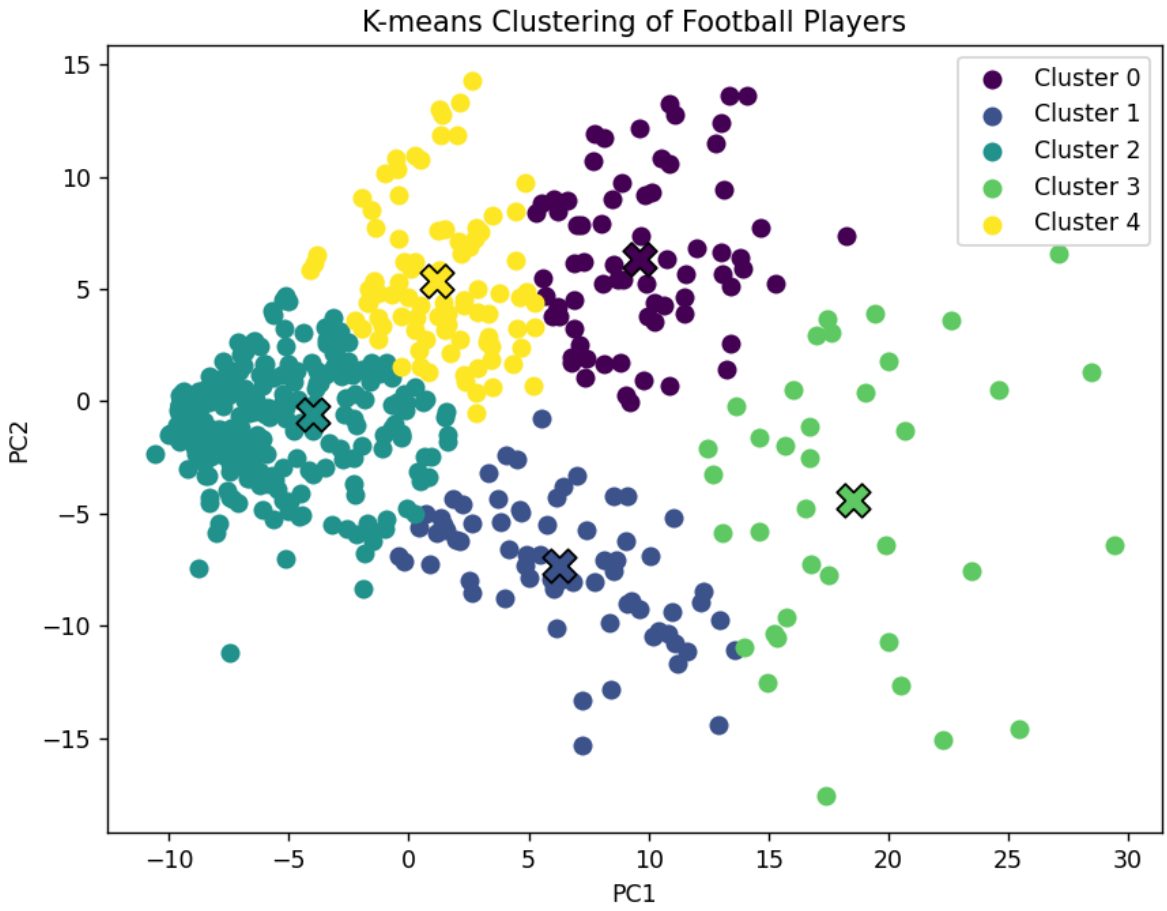
(Hình 3.1)

+ Trong hàm main , ta khởi tạo các tham số cần thiệt cho hàm vẽ biểu đồ (Hình 3.1.2)

```python
if __name__ == "__main__":
    # Đọc file csv
    data = pd.read_csv('results.csv')
    # Loại bỏ các cột ở dạng chuỗi
    data = data.select_dtypes(exclude=['object'])
    # Điền các ô N/a bằng trung bình của cột đó
    data = data.fillna(data.mean())
    # Chuẩn hóa dữ liệu
    scaler_standard = StandardScaler() # Khởi tạo
    data = pd.DataFrame(scaler_standard.fit_transform(data), columns=data.columns)
    # Áp dụng PCA giảm số chiều xuống 2
    pca = PCA(n_components=2)
    data = pca.fit_transform(data)
    data = pd.DataFrame(data, columns=['PC1', 'PC2'])
    # Số lượng cụm lấy ngẫu nhiên
    k = 5
    # Khởi tạo ngẫu nhiên các tâm cụm
    centroids = data.sample(n=k).values
    # Khởi tạo nhãn cho các điểm dữ liệu
    clusters = np.zeros(data.shape[0])
    epochs = 100
    for step in range(epochs):  # Giới hạn số bước lặp
        # Bước 1: Gán nhãn dựa trên khoảng cách đến các tâm cụm
        for i in range(len(data)):
            distances = np.linalg.norm(data.values[i] - centroids, axis=1)
            clusters[i] = np.argmin(distances)
        # Bước 2: Cập nhật các tâm cụm
        new_centroids = np.array([data.values[clusters == j].mean(axis=0) for j in range(k)])
        # Kiểm tra nếu các tâm cụm không thay đổi thì kết thúc
        if np.all(centroids == new_centroids):
            # Vẽ biểu đồ
            plot_kmeans(data.values, centroids, clusters, step)
            break
        centroids = new_centroids
```

(Hình 3.1.2)

+ Kết quả sau khi chạy code (HÌnh 3.1.3):



(Hình 3.1.3)

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

+ hàm vẽ biểu đồ (Hình 3.2.1)

```python
def radar_chart(data, player1, player2, attributes):
    # Lọc dữ liệu cho từng cầu thủ
    p1_data = data[data['Name'] == player1]
    p2_data = data[data['Name'] == player2]
    # Kiểm tra nếu không tìm thấy cầu thủ
    if p1_data.empty or p2_data.empty:
        print("Không tìm thấy cầu thủ hoặc sai tên.")
        return
    # Lấy giá trị các thuộc tính
    p1_values = p1_data[attributes].values.flatten()
    p2_values = p2_data[attributes].values.flatten()
    # Xây dựng góc của biểu đồ radar
    num_vars = len(attributes)
    angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
    # Hoàn thành vòng radar
    p1_values = np.concatenate((p1_values, [p1_values[0]]))
    p2_values = np.concatenate((p2_values, [p2_values[0]]))
    angles += angles[:1]
    # Vẽ biểu đồ
    fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
    ax.fill(angles, p1_values, color='blue', alpha=0.25)
    ax.fill(angles, p2_values, color='red', alpha=0.25)
    ax.plot(angles, p1_values, color='blue', linewidth=2, label=player1)
    ax.plot(angles, p2_values, color='red', linewidth=2, label=player2)
    # Cấu hình các nhãn thuộc tính
    ax.set_yticklabels([])
    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(attributes)
    plt.title(f"So sánh chỉ số giữa {player1} và {player2}")
    plt.legend(loc='upper right')
    plt.show()
```

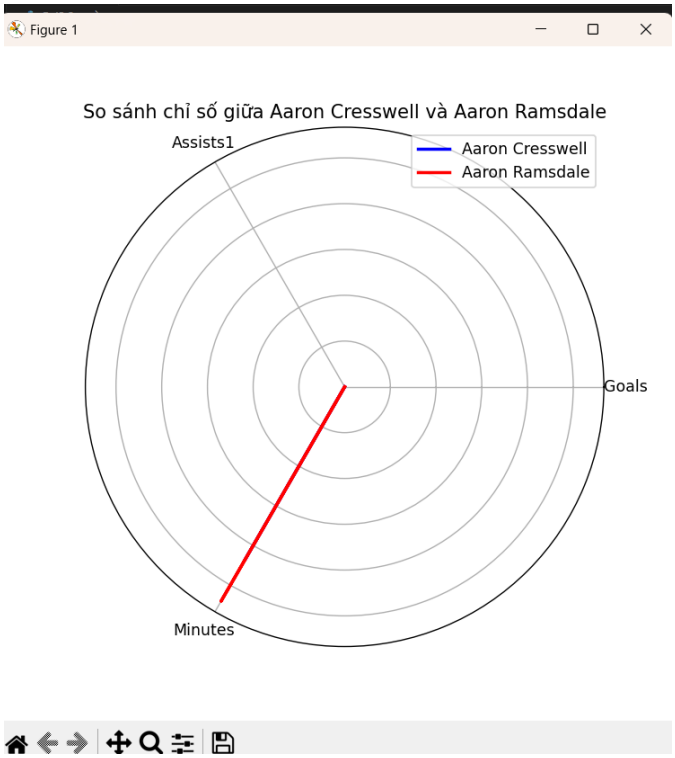
(Hình 3.2.1)

+ Hàm main (Hình 3.2.2)

```python
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Vẽ biểu đồ radar so sánh cầu thủ")
    parser.add_argument("--p1", type=str, required=True, help="Tên cầu thủ thứ nhất")
    parser.add_argument("--p2", type=str, required=True, help="Tên cầu thủ thứ hai")
    parser.add_argument("--Attribute", type=str, required=True, help="Danh sách các chỉ số cần so sánh, cách nhau bởi dấu phẩy")
    args = parser.parse_args()
    attributes = [attr.strip() for attr in args.Attribute.split(",")]
    # Đọc dữ liệu và vẽ biểu đồ
    data = load_data("results.csv")
    radar_chart(data, args.p1, args.p2, attributes)
    #C:/Users/luong/AppData/Local/Programs/Python/Python312/python.exe Bai3,2.py --p1 "Aaron Cresswell" --p2 "Aaron Ramsdale" --Attribute "Goals, Assists1, Minutes"
```

(Hình 3.2.2)

➔ Sau khi chạy lệnh ta dung lệnh <Đường dẫn file> <tên file> --p1 "Tên Cầu Thủ 1" --p2 "Tên Cầu Thủ 2" --Attribute "ChỉSố1,ChỉSố2,...,ChỉSốN"
_ Cụ thể sau khi dung lệnh :
C:/Users/luong/AppData/Local/Programs/Python/Python312/python.exe Bai3,2.py --p1 "Aaron Cresswell" --p2 "Aaron Ramsdale" --Attribute "Goals, Assists1, Minutes" (đã được bôi xanh dòng cuối cùng) trên terminal (Hình 3.3.3)



(Hình 3.3.3)

- Bài 4 : Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024 từ trang web

- Ý tưởng :
    - Cũng như bài 1 , ta dung hai thự viện requests và BeautifulSoup để thu thập và làm sạch dữ liệu trên web
    - Đầu tiên ta lấy dữ liệu của các team
    - TIếp theo , từ từng team ta lấy dữ liệu của các cầu thủ trong team đấy (GIá chuyển nhượng)
- Lấy dữ liệu từ các team (HÌnh 4.1)

```python
url = 'https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league/2023-2024'
# Cào dữ liệu từ trang web
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

table = soup.find('table', {
    'class': 'table table-striped table-hover leaguetable mvp-table ranking-table mb-0'
})
teams_data = []
if table:
    tbody = table.find('tbody')
    if tbody:
        teams = tbody.find_all('a', href=True)

        for team in teams:
            teams_data.append([team.text.strip(), team['href']])

        print ("Hoàn thành lấy dữ liệu các team")
    else:
        print('Không tìm thấy tbody')
else:
    print('Không tìm thấy bảng dữ liệu')
players_data = []
```

(Hình 4.1)

- Tiếp đó ta cào dữ liệu từng team như là câu 1 r đưa chỉ số tên và giá chuyển nhượng vào list player_data

```python
players_data = []
for team in teams_data:
    team_name = team[0]
    team_url = team[1]
    # print(team_name, team_url)

    r_tmp = requests.get(team_url)
    soup_tmp = BeautifulSoup(r_tmp.text, 'html.parser')

    table_tmp = soup_tmp.find('table', {
        'class': 'table table-striped-rowspan ft-table mb-0'
    })

    if  table_tmp:
        tbody_tmp = table_tmp.find('tbody')
        if tbody_tmp:
            players = tbody_tmp.find_all('tr')

            for player in players:
                if "odd" in player['class'] or "even" in player['class']:
                    player_name = player.find('th').find('span').get_text(strip = True)
                    player_cost = player.find_all('td')[-1].get_text(strip = True)
                    players_data.append([player_name, team_name,  player_cost])

            print ("<------------Hoàn thành lấy giá các cầu thủ của team: ", team_name, "---------------->")
        else:
            print('Không tìm thấy tbody cầu thủ')
    else:
        print('Không tìm thấy bảng dữ liệu cầu thủ')

    time.sleep(3)
```

(Hình 4.2)

- Cuối cùng , chuyển list ơlayer_data thành dataframe và đưa vào file results4.csv (HÌnh 4.3)

```
df = pd.DataFrame(players_data, columns=['Player', 'Team', 'Cost'])
df.to_csv("results4.csv", index=False, encoding='utf-8-sig')
# print(df)
print("<-----------------Đã lưu thông tin  giá các cầu thủ vào file results4.csv-------------------->")
```

- Đề xuất phương pháp định giá cầu thủ.