HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN LẬP TRÌNH PYTHON

BÁO CÁO BÀI TẬP LỚN PYTHON

Giảng viên hướng dẫn: Kim Ngọc Bách

Sinh viên thực hiện: Phạm Tiến Nghĩa

Mã sinh viên: B22DCKH083

Mã nhóm: 11

Hà Nội, tháng 11 năm 2024

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.

***Thư viện cần cài: BeautifulSoup, requests, pandas**

Cách cài: vào CMD gõ “pip install BeautifulSoup” và nhấn ENTER

Cách cài: vào CMD gõ “pip install requests” và nhấn ENTER

Cách cài: vào CMD gõ “pip install pandas” và nhấn ENTER

Ý TƯỞNG:Sử dụng thư viện requests để gửi yêu cầu để truy cập vào URL của trang Web và lấy về nội dung HTML của trang đó. Sau đó, dùng thư viện BeautifulSoup để phân tích và truy xuất các thẻ <table> chứa dữ liệu cần sử dụng.

TRONG CODE:

- Đầu tiên lấy và xử lý HTML của trang Ngoại Hạng Anh 2023-2024 thông qua URL bằng thư viện requests, dùng thư viện BeautifulSoup để tìm thẻ <table> đầu tiên

```
def main():
    url = 'https://fbref.com/en/comps/9/2023-2024/2023-2024-Premier-League-Stats'

    team_data = fetch_team_data(url)
    players_data = Crawl_Data_For_Each_Team([], team_data)  # Ensure this function is defined

    # Sort players by first name and age in descending order
    players_data.sort(key=lambda x: (x[0].split()[0], -int(x[4])))

    save_players_to_csv(players_data)
```

- Sau đó, tạo một list *team_data* để chứa thông tin của các đội bóng, mỗi đội bóng sẽ có 2 thông tin là tên đội và URL của đội. Tiếp tục tìm các thẻ <a> trong thẻ <table> vừa tìm được và duyết lần lượt các thẻ <a> đó và chỉ xử lý các thẻ mà có chuỗi “squads” trong nội dung của href và lấy nội dung của thẻ đó ta được tên đội, lấy nội dung href ta được URL của đội và lưu vào list vừa tạo.Đã có danh sách thông tin cần thiết về các đội bóng

```
    team_data = []

    if table:
        tbody = table.find('tbody')
        if tbody:
            teams = tbody.find_all('a', href=True)
            for team in teams:
                if "squads" in team['href']:
                    team_name = team.get_text(strip=True)
                    team_url = "https://fbref.com" + team['href']
                    team_data.append([team_name, team_url])
            print("-----Danh sách các đội bóng đã được lấy thành công.-----")
        else:
            print("Không tìm thấy thẻ <tbody>.")
    else:
        print("Không tìm thấy thẻ <table>.")

    return team_data
```

- Trong hàm Crawl_Data_For_Each_Team, sẽ duyệt từng thông tin các đội đã lưu ở team_data và sẽ lấy URL của đội đang xử lý để lấy HTML.

  +,Sau khi lấy được HTML của đội, tạo một list player_data_tmp để lưu thông tin các cầu thủ của đội đang xử lý và một map với key là tên cầu thủ còn value sẽ chiếu đến list chứa thông tin chỉ số của cầu thủ đó.
  +,Tiếp, xử lý 10 thẻ <table> mỗi thẻ này sẽ chứa các thông tin về cầu thủ và chỉ số của họ. 10 bảng này đều có nội dung class giống nhau nhưng nội dung id khác nhau nên có thể dễ dàng tìm kiếm.
  +,Đầu tiên, xử lý bảng Standard Stats với id trong thẻ <table> là “stats_standard_9”. Tìm thẻ <tbody> chứa dữ liệu cảu bảng sau đó tìm tất cả các thẻ <tr> chứa thông tin cầu thủ. Duyệt qua các thẻ <tr> vừa tìm được, tìm thẻ <td> với “data-stat = minutes” để lấy tổng thời gian thi đấu của cầu thủ đang xét, nếu nhỏ hơn 90 phút thì bỏ qua, còn không thì gọi một hàm con đặc biệt để xử lý thông tin

```python
def Crawl_Data_For_Each_Team(players_data, team_data):
    # Lấy thông tin và các chỉ số cầu thủ của mỗi đội
    for team in team_data:
        team_name = team[0]
        team_url = team[1]

        print(f"-----Đang cào dữ liệu cầu thủ của đội {team_name}.....-----")
        # Cào url của từng đội bóng
        r_tmp = requests.get(team_url)
        soup_tmp = BeautifulSoup(r_tmp.content, 'html.parser')

        # Danh sách tạm thời chứa thông tin tất cả cầu thủ của đội bóng hiện tại
        player_data_tmp = []
        mp = {} # Map ánh xạ đến list chứa thông tin và chỉ số của cầu thủ thông qua key là tên cầu thủ

        # Tìm bảng chứa thông tin các cầu thủ
        player_table = soup_tmp.find('table', {
            'class': 'stats_table sortable min_width',
            'id': 'stats_standard_9'
        })

        if player_table:
            # Tìm thẻ <tbody> trong <table>
            tbody = player_table.find('tbody')
            if tbody:
                players = tbody.find_all('tr')
                for player in players:
                    player_minutes_matches = player.find('td', {'data-stat': 'minutes'}).get_text(strip=True) if player.find('td', {'data-stat': 'minutes'}).get_text(stri
                    # Lọc ra những cầu thủ đã thi đấu ít nhất 90 phút
                    if player_minutes_matches == "N/a" or int(player_minutes_matches.replace(',','')) < 90:
                        continue
                    player_data_tmp = Data_Processing_of_Footballer(player, team_name, player_data_tmp, mp)
            else:
                print(f"<Không tìm thấy thẻ <tbody> trong bảng cầu thủ>")
        else:
            print(f"<Không tìm thấy thẻ <table> chứa cầu thủ trong trang của đội {team_name}.>")
```

+,Trong hàm Data_Processing_of_Footballer, sẽ lấy thông tin từng chỉ số thông qua các thẻ <td> với “data-stat” riêng biệt .Trong hàm này sẽ trả về 1 list (

```python
# Hàm xử lý dữ liệu cầu thủ
def Data_Processing_of_Footballer(tmp_tr, team_name, player_data_tmp, mp):
    try:
        # Thu thập thông tin cơ bản
        player_name = tmp_tr.find('th', {'data-stat': 'player'}).get_text(strip=True)

        player_national = (
            tmp_tr.find('td', {'data-stat': 'nationality'}).find('a')['href'].split('/')[-1].replace('-Football', ' ')
            if tmp_tr.find('td', {'data-stat': 'nationality'}).find('a')
            else "N/a"
        )
        player_position = tmp_tr.find('td', {'data-stat': 'position'}).get_text(strip=True)
        player_age = tmp_tr.find('td', {'data-stat': 'age'}).get_text(strip=True)

        # Thông tin về thời gian thi đấu
        player_games = tmp_tr.find('td', {'data-stat': 'games'}).get_text(strip=True) or "N/a"
        player_games_starts = tmp_tr.find('td', {'data-stat': 'games_starts'}).get_text(strip=True) or "N/a"
        player_minutes = tmp_tr.find('td', {'data-stat': 'minutes'}).get_text(strip=True) or "N/a"

        # Hiệu suất
        player_goals_pens = tmp_tr.find('td', {'data-stat': 'goals_pens'}).get_text(strip=True) or "N/a"
        player_pens_made = tmp_tr.find('td', {'data-stat': 'pens_made'}).get_text(strip=True) or "N/a"
        player_assists = tmp_tr.find('td', {'data-stat': 'assists'}).get_text(strip=True) or "N/a"
        player_cards_yellow = tmp_tr.find('td', {'data-stat': 'cards_yellow'}).get_text(strip=True) or "N/a"
        player_cards_red = tmp_tr.find('td', {'data-stat': 'cards_red'}).get_text(strip=True) or "N/a"
```

```
    # Tạo danh sách thông tin cầu thủ
    player_info = [
        player_name, player_national, team_name, player_position, player_age,
        player_games, player_games_starts, player_minutes, player_goals_pens,
        player_pens_made, player_assists, player_cards_yellow, player_cards_red,
        player_xg, player_npxg, player_xg_assist, player_progressive_carries,
        player_progressive_passes, player_progressive_passes_received,
        player_goals_per90, player_assists_per90, player_goals_assists_per90,
        player_goals_pens_per90, player_goals_assists_pens_per90, player_xg_per90,
        player_xg_assist_per90, player_xg_xg_assist_per90, player_npxg_per90,
        player_npxg_xg_assist_per90
    ]
```

+,Các bảng còn lại(9) sẽ xử lý giống với bảng Goalkeeping, mỗi bảng có một hàm xử lý thông tin riêng biệt

+,Sau khi xử lý xong 10 bảng, ta sẽ thêm list player_data_tmp vào players_data và sẽ tạm dừng khoảng 5 giây để xử lý đội tiếp theo (tránh bị web chặn).

```
    # Thêm dữ liệu các cầu thủ của đội bóng vào danh sách chứa dữ liệu của tất cả các cầu thủ
    players_data += player_data_tmp
    print(f"<<<<<<<<Đã cào xong dữ liệu cầu thủ của đội {team_name}.>>>>>>>")

    # Tạm nghỉ trước khi cào đội tiếp theo
    time.sleep(5)

return players_data
```

- Sau khi đã xử lý và lấy thông tin các cầu thủ của các đội, ta sẽ sort players_data theo thứ tự từ điển của first name và tuổi giảm dần(nếu first name bằng nhau). Tiếp đó, dùng thư viện pandas để tạo một DataFrame từ players_data và thêm tên cho các cột chỉ số, thông tin rồi lưu vào file “results.csv” thông qua hàm của thư viện pandas.

```python
def save_players_to_csv(players_data, filename="results.csv"):
    df_players = pd.DataFrame(players_data, columns=[
        "Player Name", "Nation", "Team", "Position", "Age", "Matches Played",
        "Starts", "Minutes", "Non-Penalty Goals", "Penalties Made", "Assists",
        "Yellow Cards", "Red Cards", "xG", "npxG", "xAG", "PrgC", "PrgP", "PrgR",
        "Gls/90", "Ast/90", "G+A/90", "G-PK/90", "G+A-PK/90", "xG/90", "xAG/90",
        "xG+xAG/90", "npxG/90", "npxG+xAG/90", "Goalkeeping_GA", "Goalkeeping_GA90",
        "Goalkeeping_SoTA", "Goalkeeping_Saves", "Goalkeeping_Save%", "Goalkeeping_W",
        "Goalkeeping_D", "Goalkeeping_L", "Goalkeeping_CS", "Goalkeeping_CS%",
        "Goalkeeping_PKatt", "Goalkeeping_PKA", "Goalkeeping_Pksv", "Goalkeeping_PKm",
        "Goalkeeping_Save%", "Shooting_Gls", "Shooting_Sh", "Shooting_SoT",
        "Shooting_SoT%", "Shooting_Sh/90", "Shooting_SoT/90", "Shooting_G/Sh",
        "Shooting_G/SoT", "Shooting_Dist", "Shooting_FK", "Shooting_PK",
        "Shooting_PKatt", "Shooting_xG", "Shooting_npxG", "Shooting_npxG/Sh",
        "Shooting_G-xG", "Shooting_np:G-xG", "Passing_Cmp", "Passing_Att",
        "Passing_Cmp%", "Passing_ToDist", "Passing_PrgDist", "Passing_Short_Cmp",
        "Passing_Short_Att", "Passing_Short_Cmp%", "Passing_Med_Cmp",
        "Passing_Med_Att", "Passing_Med_Cmp%", "Passing_Long_Cmp",
        "Passing_Long_Att", "Passing_Long_Cmp%", "Passing_Ast", "Passing_xAG",
        "Passing_xA", "Passing_A-xAG", "Passing_KP", "Passing_1/3", "Passing_PPA",
        "Passing_CrsPA", "Passing_PrgP", "Pass_Types_Live", "Pass_Types_Dead",
        "Pass_Types_FK", "Pass_Types_TB", "Pass_Types_Sw", "Pass_Types_Crs",
        "Pass_Types_TI", "Pass_Types_CK", "Pass_Types_In", "Pass_Types_Out",
        "Pass_Types_Str", "Pass_Types_Gmp", "Pass_Types_Off", "Pass_Types_Blocks",
        "GSCreation_SCA", "GSCreation_SCA90", "GGSCreation_SCA_PassLive",
        "GSCreation_SCA_PassDead", "GSCreation_SCA_TO", "GSCreation_SCA_Sh",
        "GSCreation_SCA_Fld", "GSCreation_SCA_Def", "GSCreation_GCA",
        "GSCreation_GCA90", "GSCreation_GCA_PassLive", "GSCreation_GCA_PassDead",
        "GSCreation_GCA_TO", "GSCreation_GCA_Sh", "GSCreation_GCA_Fld",
        "GSCreation_GCA_Def", "DActions_Tkl", "DActions_TklW", "DActions_Def3rd",
        "DActions_Mid3rd", "DActions_Att3rd", "DActions_Challenges_Tkl",
        "DActions_Challenges_Att", "DActions_Challenges_Tkl%", "DActions_Challenges_Lost",
        "DActions_Blocks", "DActions_Blocks_Sh", "DActions_Blocks_Pass",
        "DActions_Int", "DActions_Tkl+Int", "DActions_Clr", "DActions_Err",
        "Possession_Touches", "Possession_Def Pen", "Possession_Def 3rd",
        "Possession_Mid 3rd", "Possession_Att 3rd", "Possession_Att Pen",
        "Possession_Live", "Possession_Att", "Possession_Succ", "Possession_Succ%",
        "Possession_Tkld", "Possession_Tkld%", "Possession_Carries",
        "Possession_TotDist", "Possession_PrgDist", "Possession_PrgC",
        "Possession_1/3", "Possession_CPA", "Possession_Mis", "Possession_Dis",
        "Possession_Rec", "Possession_PrgR", "PTime_Starts", "PTime_Mn/Start",
        "PTime_Compl", "PTime_Subs", "PTime_Mn/Sub", "PTime_unSub", "PTime_PPM",
        "PTime_onG", "PTime_onGA", "PTime_onxG", "PTime_onxGA", "MStats_Fls",
        "MStats_Fld", "MStats_Off", "MStats_Crs", "MStats_OG", "MStats_Recov",
        "MStats_Won", "MStats_Lost", "MStats_Won%"
    ])
```

CÂU 2: Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số. Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ trong toàn giải và của mỗi đội. Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội. Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024.

***Thư viện cần cài: pandas, tabulate (vẽ bảng đẹp), matplotlib, seaborn.**

Cách cài: vào CMD gõ " pip install pandas" và ấn ENTER

Cách cài: vào CMD gõ " pip install tabulate" và ấn ENTER

Cách cài: vào CMD gõ " pip install matplotlib" và ấn ENTER

Cách cài: vào CMD gõ " pip install seaborn" và ấn ENTER

Ý TƯỞNG:

- **Dùng pandas, dùng để xử lý và thao tác với dữ liệu dạng bảng thông qua dữ liệu lấy từ file csv "resulsts.csv" ở câu 1.**
- **Dùng tabulate, giúp định dạng dữ liệu thành bảng một cách dễ nhìn, tiện lợi khi hiển thị kết quả trong terminal.**
- **Dùng matplotlib, thư viện chính dùng để vẽ biểu đồ trong Python, còn *seaborn* được xây dựng trên *matplotlib*, cung cấp các hàm trực quan hóa dữ liệu tốt hơn và giúp biểu đồ dễ nhìn hơn.**
- **Dùng Collections để đếm tần xuất các giá trị trong tập dữ liệu ở ý cuối, thư viện os để làm việc với hệ thống tệp và thư mục, thư viện time để quản lý thời gian.**

TRONG CODE:

- Đọc file csv "results.csv" và chỉ lấy từ cột chỉ số thứ 4 trở đi. Chuyển kiểu dữ liệu với các giá trị "N/a" thành NaN dễ dàng xử lý và các giá trị khác "N/a" thành số vì file csv tất cả giá trị ở dạng chuỗi.

```
if __name__ == "__main__":
    df = pd.read_csv("results.csv")
    columns_to_analyze = df.columns[4:]  # Các cột chỉ số

    df[columns_to_analyze] = df[columns_to_analyze].apply(pd.to_numeric, errors='coerce')
```

- Cung cấp cho người dùng 5 chức năng, trong đó có 5 chức năng ứng với từng yêu cầu đề đề bài và thoát.

```
if __name__ == "__main__":
    df = pd.read_csv("results.csv")
    columns_to_analyze = df.columns[4:]  # Các cột chỉ số

    df[columns_to_analyze] = df[columns_to_analyze].apply(pd.to_numeric, errors='coerce')

    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")
    print("3. Vẽ biểu đồ histogram")
    print("4. Tìm đội có giá trị cao nhất và đếm tần suất")
    print("5. Thoát chương trình")

    while True:
        try:
            choice = int(input("Nhập lựa chọn của bạn: "))
            if choice == 1:
                export_top_players(df, columns_to_analyze)
            elif choice == 2:
                calculate_statistics(df, columns_to_analyze)
            elif choice == 3:
                plot_historgrams(df, columns_to_analyze)
            elif choice == 4:
                find_best_team(df, columns_to_analyze)
            elif choice == 5:
                print("Thoát chương trình")
                break
            else:
                print("Lựa chọn không hợp lệ, vui lòng chọn lại.")
        except ValueError:
            print("Vui lòng nhập một số.")
```

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

- Xử lý trong hàm get_top3
- Dùng nlargest,nsmallest để tìm 3 cầu thủ có chỉ số cao nhất, đung thư viện tabulate để vẽ bảng dễ nhìn hơn (Hình 2.3) và ghi vào file Top3NguoiChiSoCaoNhat.txt và Top3NguoiChiSoThapNhat.txt

```
def export_top_players(df, columns_to_analyze, n=3):
    """
    Lưu vào file các cầu thủ có chỉ số cao nhất và thấp nhất cho các cột chỉ định.
    """
    # Xuất top 3 cầu thủ có chỉ số cao nhất và thấp nhất
    for top_type, func, filename in [("cao nhất", "nlargest", "Top3NguoiChiSoCaoNhat.txt"),
                                     ("thấp nhất", "nsmallest", "Top3NguoiChiSoThapNhat.txt")]:
        with open(filename, "w", encoding="utf-8") as file:
            for column in columns_to_analyze:
                file.write(f"\nTop {n} cầu thủ {top_type} cho chỉ số '{column}':\n")
                top_players = getattr(df, func)(n, column)[['Player Name', 'Team', column]]
                file.write(tabulate(top_players, headers='keys', tablefmt='fancy_grid') + "\n")
            print(f"<<<<<<Đã ghi kết quả {top_type} vào file {filename}>>>>>>")
```

-

2. 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.

   -DataFrame chứa dữ liệu cần phân tích

   -columns_to_analyze: Danh sách các cột trong DataFrame mà bạn muốn tính toán thống kê.

   -Tạo một DataFrame mới gọi là overall_stats với một hàng cho toàn giải ('Team': ['All']).

   -Sử dụng comprehension dictionary để tính toán các thống kê cho từng cột trong columns_to_analyze.

   - Getattr(df[col], stat)() lấy phương thức tương ứng (median, mean, std) từ cột và tính toán giá trị, sau đó làm tròn đến 2 chữ số thập phân.

   -Nhóm dữ liệu theo cột 'Team' và áp dụng các hàm median, mean, std cho các cột trong columns_to_analyze.

   -reset_index() được gọi để biến chỉ số nhóm thành cột, giúp dễ dàng làm việc với DataFrame.

   - Đổi tên các cột trong team_stats để dễ hiểu hơn, theo định dạng "Tên cột thống kê của cột dữ liệu"

   -Sử dụng pd.concat để kết hợp thống kê toàn giải và thống kê cho từng đội thành một DataFrame duy nhất (final_stats).

   - Xuất DataFrame này ra file CSV có tên results2.csv, không bao gồm chỉ số hàng và sử dụng mã hóa UTF-8 với BOM.

   -In ra thông báo xác nhận rằng kết quả đã được xuất ra file.

```python
def calculate_statistics(df, columns_to_analyze):
    """
    Tính và xuất thống kê (median, mean, std) của các chỉ số cho toàn giải và từng đội.
    """
    # Tính toán toàn giải
    overall_stats = pd.DataFrame({
        'Team': ['All'],
        **{f"{stat.capitalize()} of {col}": getattr(df[col], stat)().round(2) for col in columns_to_analyze for stat in ['median', 'mean', 'std']}
    })

    # Tính toán cho từng đội
    team_stats = df.groupby('Team')[columns_to_analyze].agg(['median', 'mean', 'std']).round(2).reset_index()
    team_stats.columns = ['Team'] + [f"{stat.capitalize()} of {col}" for col in columns_to_analyze for stat in ['median', 'mean', 'std']]

    # Kết hợp và lưu file CSV
    final_stats = pd.concat([overall_stats, team_stats], ignore_index=True)
    final_stats.to_csv('results2.csv', index=False, encoding='utf-8-sig')
    print("<<<<<<<<Đã xuất kết quả ra file results2.csv>>>>>>>>")
```

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

- Xử lý trong hàm print_historgram.
- Vì số lượng biểu đồ rất nhiều nên sẽ tạo một folder (nếu không tồn tại) để chứa các biểu đồ chỉ số của toàn giải, tạo một folder (nếu không tồn tại) để chứa các biểu đồ chỉ số cảu từng đội bóng (trong folder này sẽ chia thành nhiều folder con có tên là tên đội bóng để lưu các biểu đồ của đội bóng đó). Xem code để rõ hơn về cái này.
- Với toàn giải, thì sử dụng thư viện matplotlib vs seaborn để vẽ với các thông số biểu đồ đã được tạo trong code.

```python
# Vẽ histogram cho toàn giải
for col in columns_to_analyze:
    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")
```

- Với từng đội, vẫn dùng 2 thư viện như trên để vẽ. Trước tiên phải lấy danh sách các đội bóng khác nhau thông qua hàm unique(). Duyệt lần lượt các đội và vẽ các biểu đồ. Dừng lại 3 giây sau đó mới vẽ tiếp đội khác.

```python
# 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 columns_to_analyze:
        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>>>>>>>>>>")
```

3. Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024**.**

   Hàm `find_best_team` được thiết kế để xác định đội bóng có giá trị cao nhất trong từng chỉ số đã chỉ định và đếm số lần mà mỗi đội xuất hiện là đội tốt nhất.

   Hàm nhận vào hai tham số: `df`, một DataFrame chứa dữ liệu các đội bóng, và `columns_to_analyze`, danh sách các cột chỉ số cần phân tích.

   Đầu tiên, hàm nhóm dữ liệu theo tên đội và tính giá trị trung bình cho các chỉ số này. Sau đó, nó tìm ra đội có giá trị cao nhất cho từng chỉ số bằng cách sử dụng phương thức `idxmax()` để lấy tên đội và `max()` để có giá trị tương ứng.

   Kết quả đội tốt nhất cho từng chỉ số được in ra dưới dạng bảng. Tiếp theo, hàm đếm tần suất xuất hiện của các đội bằng cách sử dụng `Counter` và sắp xếp tần suất từ cao đến thấp.

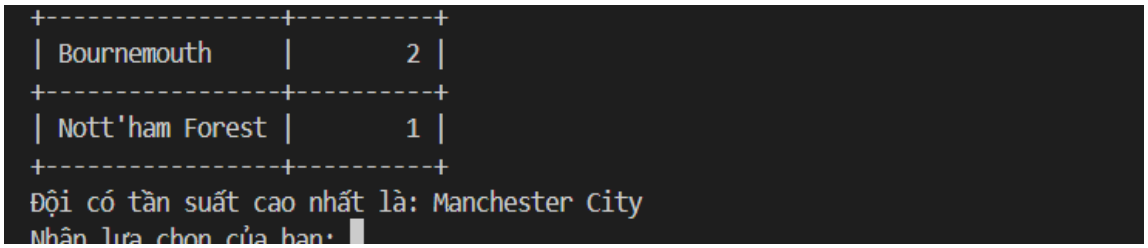
   Cuối cùng, tần suất của từng đội cũng được in ra dưới dạng bảng, cùng với tên đội có tần suất cao nhất. Hàm này cung cấp cái nhìn tổng quan về hiệu suất của các đội bóng trong một giải đấu, giúp phân tích và đánh giá thành tích của họ.

```python
def find_best_team(df, columns_to_analyze):
    """
    Tìm đội có giá trị cao nhất ở từng chỉ số và đếm tần suất.
    """
    team_summary = df.groupby('Team')[columns_to_analyze].mean()
    best_teams = [(col, team_summary[col].idxmax(), team_summary[col].max()) for col in columns_to_analyze]

    # In kết quả
    print(tabulate(best_teams, headers=["Chỉ số", "Team", "Giá trị"], tablefmt="grid"))

    # Đếm tần suất
    team_counts = Counter([item[1] for item in best_teams])
    frequency_table = sorted(team_counts.items(), key=lambda x: x[1], reverse=True)

    # In tần suất
    print("\nTần suất của từng đội:")
    print(tabulate(frequency_table, headers=["Team", "Số lần"], tablefmt="grid"))
    print(f"Đội có tần suất cao nhất là: {frequency_table[0][0]}")
```

```
Nhập lựa chọn của bạn: 4
+--------------------------+-----------------+--------------+
| Chỉ số                   | Team            |      Giá trị |
+==========================+=================+==============+
| Age                      | West Ham        |    28.2727   |
+--------------------------+-----------------+--------------+
| Matches Played           | Fulham          |    27.2381   |
+--------------------------+-----------------+--------------+
| Starts                   | Fulham          |    19.9048   |
+--------------------------+-----------------+--------------+
| Minutes                  | Liverpool       |   778.25     |
+--------------------------+-----------------+--------------+
| Non-Penalty Goals        | Manchester City |     4.04762  |
+--------------------------+-----------------+--------------+
```

Tần suất của từng đội:

Team	Số lần
Manchester City	54
Liverpool	32
Fulham	15
Arsenal	13



Đội có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024:**Manchester City**

CÂU 3: **Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau. Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? Bạn có Nhận xét gì về kết quả. Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D. Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ**.

***Thư viện cần cài: pandas, sklearn, numpy, matplotlib**

Cách cài: vào CMD và gõ "pip install pandas numpy sklearn matplotlib" và ấn ENTER

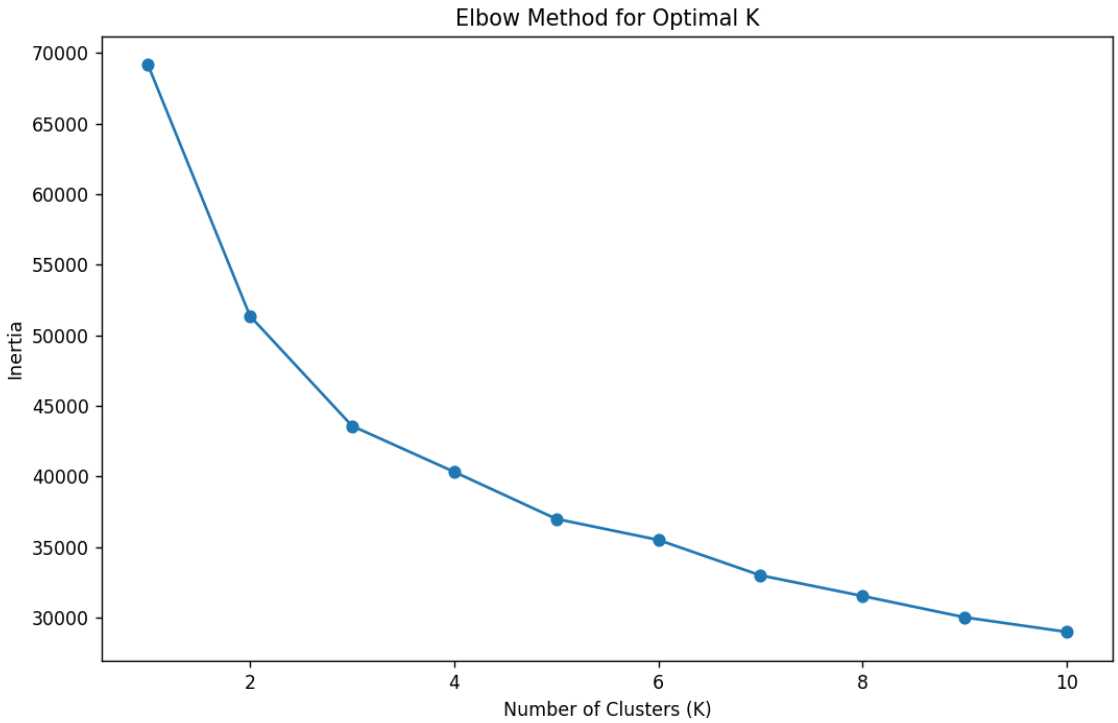
Ý TƯỞNG:

- **Dùng pandas để đọc file csv chứa dữ liệu và xử lý dữ liệu trước khi dùng thuật toán**
- **Dùng StandardScaler từ sklearn.preprocessing để chuẩn hoá dữ liệu để mỗi đặc trưng có phương sai bằng 0 và độ lệch bằng 1, giúp cải thiện hiệu suất của K-means.**
- **Dùng PCA từ sklearn.decomposition: Giảm số chiều của dữ liệu để dễ trực quan hóa.**
- **Dùng numpy để tính toán số học và các thao tác mảng để thực hiện các bước của K-means, như tính khoảng cách Euclidean và trung bình của các điểm trong cụm.**
- **Dùng matplotlib để vẽ biểu đồ.**
- **Để phân loại cầu thủ thành số nhóm phù hợp thì dùng phương pháp Elbow.**
- **Dùng argparse để phân tích các đối số dòng lệnh (command-line arguments), cho phép chương trình nhận các tham số từ người dùng khi chạy từ dòng lệnh (ý so sánh hai cầu thủ).**

TRONG CODE:

1. Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau. Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? Bạn có Nhận xét gì về kết quả. Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.

- Lựa chọn số lượng nhóm phù hợp thì dùng phương pháp Elbow



Điểm khuỷu tay (Elbow point): Quan sát đồ thị, chúng ta tìm điểm mà sau đó đường cong bắt đầu giảm dần với tốc độ chậm hơn. Điểm này thường được gọi là "điểm khuỷu tay". Giá trị K tương ứng với điểm khuỷu tay thường được chọn làm số lượng cụm tối ưu. Trong đồ thị bạn cung cấp, ta thấy đường cong giảm khá nhanh từ K=1 đến K=4, sau đó giảm dần với tốc độ chậm hơn**.** Điểm khuỷu tay có thể xem là nằm ở khoảng K=3.
Tiếp theo, sử dung pandas để đọc file csv chứa dữ liệu và xử lý các dữ liệu trước khi đưa vào chuẩn hoá

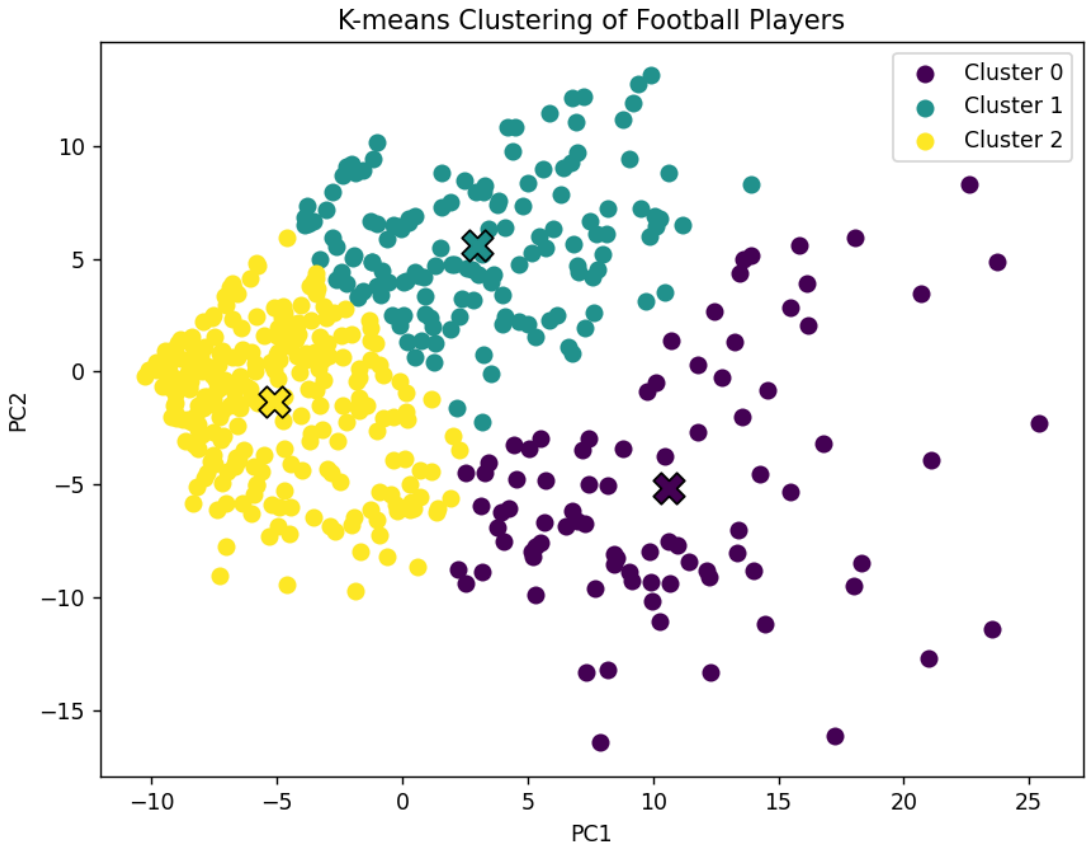
```
if __name__ == "__main__":
    # Đọc file csv
    data = pd.read_csv('results.csv')

    # 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 hoá dữ liệu bằng StandardScaler rồi dùng PCA giảm số chiều xuống 2**.**

```
# 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'])
```

- Sau đó dùng K-means để phân loại cầu thủ với số lượng cụm (nhóm) K = 3 đã suy ra từ phương pháp Silhouette Scor và sẽ tạo ngẫu nhiên K tâm cụm. Nội dung đoạn code K-means là sẽ tính khoảng các từ các điểm đến K tâm cụm nếu gần với tâm cụm nào nhất thì sẽ đánh theo màu tâm cụm đó, sau đó lấy trung bình toạ độ của các điểm cùng màu để cập nhập tâm cụm mới và cứ tiếp tục lặp lại như thế đến khi nào tâm cụm sau khi cập nhập không thay đổi với trước khi cập nhập thì dừng và gọi hàm vẽ biểu đồ (xem code của hàm trong file Bai3-1.py).

```
# K-means
# Số lượng cụm
k = 3

# 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
```



1. Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ. Với đầu vào như sau:

   + python radarChartPlot.py --p1 <player Name 1> --p2 <player Name 2> --Attribute <att1,att2,...,att_n>
   + --p1: là tên cầu thủ thứ nhất
   + --p2: là tên cầu thủ thứ hai
   + --Attribute: là danh sách các chỉ số cần so sánh

   - Khởi tạo parper để lấy thông số đầu vào.

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Compare players using radar chart.')
    parser.add_argument('--p1', type=str, required=True, help='Name of player 1')
    parser.add_argument('--p2', type=str, required=True, help='Name of player 2')
    parser.add_argument('--Attribute', type=str, required=True, help='Comma-separated list of attributes')
```

   - Đọc dữ liệu từ file csv bằng pandas và xử lý các dữ liệu về dạng số (Vì dữ liệu đang ở dạng chuỗi). Gọi hàm để vẽ rada (plot_radar_chart).

```
args = parser.parse_args()

# Đọc dữ liệu từ file CSV
data = pd.read_csv('results.csv')  # Thay 'results.csv' bằng tên file của bạn

# Lấy danh sách các chỉ số từ đối số
attributes = args.Attribute.split(',')
data[attributes] = data[attributes].apply(pd.to_numeric, errors='coerce') # Vì các giá trị trong file csv ở dạng String nên phải ép kiểu


# Gọi hàm vẽ biểu đồ radar
radar_chart(data, args.p1, args.p2, attributes)
```

- Ở hàm radar_chart, trích xuất dữ liệu của hai cầu thủ từ data dựa trên tên đã cung cấp (player1 và player2) và các thuộc tính (attributes). Xác định số lượng thuộc tính (num_vars) để tính toán các góc cho biểu đồ radar.

```
def radar_chart(data, player1, player2, attributes):
    # Số lượng chỉ số
    num_vars = len(attributes)

    # Tạo một mảng cho mỗi cầu thủ
    player1_values = data[data['Player Name'] == player1][attributes].values.flatten()
    player2_values = data[data['Player Name'] == player2][attributes].values.flatten()

    # Tạo một mảng góc cho các chỉ số
    angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
```

- Thiết lập chỉ số, thuộc tính để vẽ biểu đồ rada (xem chi tiết hơn trong file Bai3-2.py**).**

  Cách chạy file Cau3-2.py để so sánh 2 cầu thủ: save file trước -> vào terminal gõ theo format này: python bai3-2.py –p1 “tên cầu thủ 1” –p2 “tên cầu thủ 2” –Attribute “các thuộc tính cần so sánh (sử dụng dấu phẩy để phân cách)” -> Nhấn ENTER để chạy
- Ví dụ: python bai3-2.py --p1 "Alejandro Garnacho" --p2 "Alexander Isak" --Attribute "Starts,Minutes,Non-Penalty Goals,Penalties Made,Assists,Yellow Cards"

Biểu đồ rada sẽ được như sau: