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: Bùi Đức Hưng

Mã sinh viên: B22DCKH057

Mã nhóm: 11

Hà Nội, ngày 3 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 sử dụng**: BeautifulSoup, requests

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

***Thực hiện**:

- Đầu tiên lấy và xử lý HTML của trang Ngoại Hạng Anh 2023-2024 thông qua URL
  - → Sử dụng requests.get(url) để tải nội dung HTML của trang web và lưu vào biến r.
  - → Khởi tạo BeautifulSoup với r.content và parser 'html.parser' để phân tích nội dung HTML, lưu vào biến soup.
  - → Tìm bảng chứa thông tin đội bóng bằng cách sử dụng soup.find('table', ...), dựa trên thuộc tính class và id để xác định đúng bảng cần tìm.

```
# URL
url = 'https://fbref.com/en/comps/9/2023-2024/2023-2024-Premier-League-Stats'
r = requests.get(url)
soup = BeautifulSoup(r.content, 'html.parser')

# Tìm bảng chứa thông tin các đội bóng
table = soup.find('table', {
    'class': 'stats_table sortable min_width force_mobilize',
    'id': 'results2023-202491_overall'
})
```

- Tạo một list *team_data* để chứa thông tin của các đội bóng gồm có 2 thông tin là tên đội và URL của đội →Tìm các thẻ <a> trong thẻ <table> vừa tìm được và chỉ xử lý các thẻ mà có chuỗi “squads” trong nội dung của href .

```
team_data = []

if table:
    # Tìm thẻ <tbody>
    tbody = table.find('tbody')
    if tbody:
        # Lấy tất cả các thẻ <a> có định dạng như yêu cầu
        teams = tbody.find_all('a', href=True)

        for team in teams:
            if "squads" in team['href']:  # Kiểm tra nếu "squads" có trong 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>.")
```

- Tạo một list *players_data* để chứa thông tin toàn bộ cầu thủ trong giải và gọi hàm Crawl_Data.

```
# # Danh sách chứa từng cầu thủ của đội bóng
players_data = []
players_data = Crawl_Data(players_data, team_data)
```

- Trong hàm Crawl_Data, 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 dictionary với tên cầu thủ là key value giá trị là danh sách chứa thông tin thống kê của cầu thủ đó.
  - Hàm sau đó sẽ duyệt qua 10 bảng <table>, mỗi bảng chứa các chỉ số khác nhau về cầu thủ. Các bảng này đều có chung một lớp CSS nhưng có id khác nhau, giúp việc truy xuất theo id trở nên thuận tiện.
  - Đầu tiên, hàm xử lý bảng Standard Stats, được nhận diện bằng id "stats_standard_9". Trong bảng này, hàm tìm thẻ <tbody> chứa dữ liệu, rồi truy xuất tất cả các thẻ <tr> đại diện cho từng 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 thời gian thi đấu nhỏ hơn 90 phút thì bỏ qua, còn không thì gọi một hàm con đặc biệt để xử lý thông tin (sẽ trả về 1 list).

```
# 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 cầu thủ , chỉ số là key

# 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-st
            # 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"

→ Chuẩn hóa dữ liệu: Định dạng tên, chuyển đổi số liệu, …và thay thế giá trị thiếu bằng "N/a"

→ Sử dụng try-except giúp chương trình của bạn chạy ổn định hơn và xử lý các trường hợp ngoại lệ một cách có kiểm soát

→Lưu trong dictionary

```python
def Data_Processing_of_Footballer(tmp_tr, team_name, player_data_tmp,
    try:
        player_xg_assist_per90 = tmp_tr.find('td', {'data-stat': 'xg_assist_per90'}).get_text(strip=True) or "N/a"
        player_xg_xg_assist_per90 = tmp_tr.find('td', {'data-stat': 'xg_xg_assist_per90'}).get_text(strip=True) or "N/a"
        player_npxg_per90 = tmp_tr.find('td', {'data-stat': 'npxg_per90'}).get_text(strip=True) or "N/a"
        player_npxg_xg_assist_per90 = tmp_tr.find('td', {'data-stat': 'npxg_xg_assist_per90'}).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
        ]

        # Lưu dữ liệu
        player_data_tmp.append(player_info)
        mp[player_name] = player_info

    except Exception as e:
        print(f"Lỗi khi xử lý cầu thủ: {e}")
```

- Thứ hai, xử lý bảng Goalkeeping với id trong thẻ <table> là “stats_keeper_9”. tam thời tên các thủ môn.
    → Tìm <tbody> trong bảng và tất cả thẻ <tr> (các cầu thủ).
    → Tạo danh sách list_tmp để lưu tên thủ môn.
    → Duyệt từng cầu thủ, lấy tên, gọi hàm Data_Processing_of_Goalkeeper để xử lý chỉ số, sau đó cập nhật dữ liệu vào mp và thêm tên vào list_tmp.
    ➔ Với các cầu thủ không có trong list_tmp, thêm "N/a" cho 15 chỉ số

```python
Goalkeeper_table = soup_tmp.find('table', {
    'class': 'stats_table sortable min_width',
    'id': 'stats_keeper_9'
})
if Goalkeeper_table:
    # Tìm thẻ <tbody> trong <table>
    tbody = Goalkeeper_table.find('tbody')
    if tbody:
        players = tbody.find_all('tr')
        # Danh sách lưu tạm thời tên các thủ môn
        list_tmp = []

        for player in players:
            player_name = player.find('th', {'data-stat': 'player'}).get_text(strip=True)
            if player_name in mp:
                mp[player_name] += Data_Processing_of_Goalkeeper(player)
                list_tmp.append(player_name)

        for player in player_data_tmp:
            if player[0] not in list_tmp:
                player += ["N/a"] * 15
    else:
        print(f"<Không tìm thấy thẻ <tbody> bảng thủ môn.>")
else:
    print(f"<Không tìm thấy thẻ <table> chứa thủ môn trong trang của đội {team_name}.>")
```

- 8 bảng còn lại 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
players_data += player_data_tmp
print(f"----Đã lấy xong dữ liệu cầu thủ của đội {team_name}.---")

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

- Sau khi đã xử lý và lấy thông tin các cầu thủ của các đội,
→ Sắp xếp danh sách cầu thủ: players_data được sắp xếp theo First Name và nếu trùng tên thì sắp xếp tiếp theo Age từ lớn đến nhỏ.

→ Chuyển đổi dữ liệu thành DataFrame: Dữ liệu của các cầu thủ được chuyển thành DataFrame với tên cột tương ứng với các chỉ số cần thu thập.

→ Lưu dữ liệu vào file CSV: DataFrame df_players được lưu vào file results.csv với encoding utf-8-sig để đảm bảo hiển thị đúng ký tự.

→ Thông báo hoàn thành: In ra thông báo xác nhận việc lưu dữ liệu cầu thủ vào file results.csv thành công.

```
players_data = sorted(players_data, key=lambda x: (x[0].split()[0], -int(x[4])))


# # Chuyển dữ liệu thành DataFrame và lưu thành file CSV
df_players = pd.DataFrame(players_data, columns=["Player Name", "Nation", "Team", "Position", "Age", "Matches Played", "Starts", "Minute
                                                 "Goalkeeping_GA", "GGoalkeeping_GA90", "Goalkeeping_SoTA", "Goalkeeping_Saves", "Goalke
                                                 "Shooting_Gls", "Shooting_Sh", "Shooting_SoT", "Shooting_SoT%", "Shooting_Sh/90", "Shoo
                                                 "Passing_Cmp", "Passing_Att", "Passing_Cmp%", "Passing_ToDist", "Passing_PrgDist", "Pas
                                                 "Pass_Types_Live", "Pass_Types_Dead", "Pass_Types_FK", "Pass_Types_TB", "Pass_Types_Sw"
                                                 "GSCreation_SCA", "GSCreation_SCA90", "GGSCreation_SCA_PassLive", "GSCreation_SCA_PassD
                                                 "DActions_Tkl", "DActions_TklW", "DActions_Def3rd", "DActions_Mid3rd", "DActions_Att3rd
                                                 "Possession_Touches", "Possession_Def Pen", "Possession_Def 3rd", "Possession_Mid 3rd",
                                                 "PTime_Starts", "PTime_Mn/Start", "PTime_Compl", "PTime_Subs", "PTime_Mn/Sub", "PTime_u
                                                 "MStats_Fls", "MStats_Fld", "MStats_Off", "MStats_Crs", "MStats_OG", "MStats_Recov", "M
                                                 ])
df_players.to_csv("results.csv", index=False, encoding='utf-8-sig')
print("-----------------Lưu thành công thông tin các cầu thủ vào file results.csv--------------------")
```

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 sử dụng**: pandas, tabulate (vẽ bảng đẹp), matplotlib, seaborn.

- Sử dụng thư viện pandas để xử lý và thao tác với dữ liệu bảng, dựa trên dữ liệu từ file CSV "results.csv" đã thu thập ở câu 1.
- Sử dụng tabulate để định dạng dữ liệu dưới dạng bảng, giúp hiển thị kết quả dễ đọc và rõ ràng hơn trong terminal.
- Sử dụng matplotlib làm thư viện chính để vẽ biểu đồ, kết hợp với seaborn để nâng cao tính trực quan và cải thiện giao diện biểu đồ, giúp các biểu đồ trở nên dễ nhìn và dễ phân tích.
- Sử dụng collections để đếm tần suất của các giá trị trong tập dữ liệu, os để thao tác với các tệp và thư mục trong hệ thống, và time để quản lý các thao tác liên quan đến thời gian.

***Thực hiện**:

- Đọc file csv “results.csv” Sử dụng pd.read_csv("results.csv") để tải dữ liệu từ file results.csv và lưu vào DataFrame df (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.

```python
if __name__ == "__main__":
    df = pd.read_csv("results.csv")

    columns_to_analyze = df.columns[4:]  # Chỉ chọn các cột chỉ số từ cột "Non-Penalty Goals" trở đ

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

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

Trong hàm get_top3_player

- Dùng nlargest(3, column) để tìm 3 cầu thủ có chỉ số cao nhất, đung thư viện tabulate để vẽ bảng và ghi vào file Top3NguoiChiSoCaoNhat.txt.
- Tương tự với 3 cầu thủ có chỉ số thấp nhất.

```python
def get_top3_player(df, column_to_analyze):
    # Top 3 cao nhat
    with open("Top3NguoiChiSoCaoNhat.txt", "w", encoding="utf-8") as file:
        for column in columns_to_analyze:
            file.write(f"\nTop 3 cầu thủ cao nhất cho chỉ số '{column}':\n")
            top_highest = df.nlargest(3, column)[['Player Name', 'Team', column]]
            file.write(tabulate(top_highest, headers='keys', tablefmt='fancy_grid') + "\n")

        print("----Đã ghi kết quả Top 3 cầu thủ cao nhất vào file Top3NguoiChiSoCaoNhat.txt-----")

    # Top 3 thấp nhất
    with open("Top3NguoiChiSoThapNhat.txt", "w", encoding="utf-8") as file:
        for column in columns_to_analyze:
            file.write(f"\nTop 3 cầu thủ thấp nhất cho chỉ số '{column}':\n")
            top_lowest = df.nsmallest(3, column)[['Player Name', 'Team', column]]
            file.write(tabulate(top_lowest, headers='keys', tablefmt='fancy_grid') + "\n")

        print("----Đã ghi kết quả Top 3 cầu thủ thấp nhất vào file Top3NguoiChiSoThapNhat.txt----")
```

```
Top 3 cầu thủ cao nhất cho chỉ số 'Matches Played':
╒════╤═══════════════╤═══════════════╤═══════════════╕
│    │ Player Name   │ Team          │   Matches Played │
╞════╪═══════════════╪═══════════════╪═══════════════╡
│ 33 │ André Onana   │ Manchester Utd │            38 │
├────┼───────────────┼───────────────┼───────────────┤
│ 62 │ Bernd Leno    │ Fulham        │            38 │
├────┼───────────────┼───────────────┼───────────────┤
│ 84 │ Carlton Morris │ Luton Town    │            38 │
╘════╧═══════════════╧═══════════════╧═══════════════╛
```

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.**

Trong hàm calculate_statistics:

- df[columns_to_analyze] để truy cập tất cả các cột chứa các chỉ số cần phân tíchTính trung vị bằng gọi hàm *median*, trung bình gọi hàm *mean*, độ lệch chuẩn gọi hàm *std*, làm tròn gọi hàm *round*.

```
# Tính các thống kê cho toàn giải và làm tròn đến 2 chữ số
stats_all = {
    'Median': df[columns_to_analyze].median().round(2),
    'Mean': df[columns_to_analyze].mean().round(2),
    'Std': df[columns_to_analyze].std().round(2)
}
```

- Tạo một DataFrame là overall_df để lưu trữ các giá trị thông kê của toàn giải : Cột STT có giá trị 0 đánh dấu rằng đây là dữ liệu của toàn giải, không phải của một đội cụ thể, Cột Team chứa giá trị "all" để biểu thị dữ liệu này đại diện cho toàn giải, **{f'{stat} of {col}': [values[col]] ...}: Tạo các cột cho mỗi chỉ số, chứa giá trị trung vị, trung bình, và độ lệch chuẩn từ stats_all.

```
# Tạo DataFrame cho toàn giải
overall_df = pd.DataFrame({
    'STT': [0],
    'Team': ['all'],
    **{f'{stat} of {col}': [values[col]] for stat, values in stats_all.items() for col in columns_to_analyze}
})
```

- Đối với các đội bóng thì cũng tạo một DataFrame là team_data để lưu trữ các giá trị thông kê của từng đội .

```
# Tính các thống kê cho từng đội và làm tròn đến 2 chữ số
stats_team = {
    'Median': df.groupby('Team')[columns_to_analyze].median().round(2),
    'Mean': df.groupby('Team')[columns_to_analyze].mean().round(2),
    'Std': df.groupby('Team')[columns_to_analyze].std().round(2)
}

# Tạo DataFrame cho từng đội
team_data = {
    'STT': range(1, len(stats_team['Median']) + 1),
    'Team': stats_team['Median'].index
}

for stat, values in stats_team.items():
    for col in columns_to_analyze:
        team_data[f'{stat} of {col}'] = values[col].values

team_df = pd.DataFrame(team_data)
```

- Sau đó sẽ gộp hai DataFrame là overall_df và team_df thành final_df.

```
# Gộp hai DataFrame lại thành một
final_df = pd.concat([overall_df, team_df], ignore_index=True)
```

- Xuất ra file CSV:

```
# Xuất ra file CSV
final_df.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.

- 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 của 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).
- Vẽ histogram cho từng chỉ số trên toàn giải: Với mỗi chỉ số trong columns_to_analyze, hàm vẽ histogram cho dữ liệu toàn giải với màu xanh. Biểu đồ có kích thước 8x6, được thêm đường cong KDE để thể hiện mật độ, cùng các nhãn trục x và y. Sau đó, biểu đồ được lưu vào thư mục "histograms_all".

```
# 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")
```

- Vẽ biểu đồ cho từng đội : Lấy danh sách các đội trong df, duyệt qua từng đội để tạo một thư mục riêng cho từng đội → Với mỗi chỉ số, hàm lọc dữ liệu của đội hiện tại và vẽ histogram với màu xanh lá cây, thêm đường KDE và các nhãn trục. Biểu đồ được lưu vào thư mục của đội, giúp dễ dàng so sánh phân bố chỉ số giữa các đội.

```
# 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 hoàn thành biểu đồ cho đội {team}")
    time.sleep(3)

print("----Hoàn thành vẽ biểu đồ cho toàn giải và từng đội----")
```

4. **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**.

Trong hàm get_best_team.

- Chỉ lấy cột chỉ số từ Non – Penalty Goals trở đi để dễ dàng đánh giá. Và nhóm theo tên đội dùng groupby và tính trung bình của các chỉ số

```
# Chuyển các cột chỉ số thành dạng số, thay thế các giá trị không hợp lệ bằng NaN
metric_columns = df.columns[8:]
df[metric_columns] = df[metric_columns].apply(pd.to_numeric, errors='coerce')

# Tính trung bình các chỉ số theo từng đội
team_averages = df.groupby('Team')[metric_columns].mean()

# Tạo danh sách lưu đội có giá trị cao nhất cho từng chỉ số
```

- Tạo một list là results để chứa kết quả sau khi xử lý tìm đội có giá trị cao nhất ở các chỉ số.

```
# Tạo danh sách lưu đội có giá trị cao nhất cho từng chỉ số
results = [
    [col, team_averages[col].idxmax(), team_averages[col].max()]
    for col in metric_columns
]

# In bảng kết quả đội có chỉ số cao nhất cho từng chỉ số
print("Đội có giá trị cao nhất cho từng chỉ số:")
print(tabulate(results, headers=["Chỉ số", "Team", "Giá trị"], tablefmt="grid"))
```

Chỉ số	Team	Giá trị
Non-Penalty Goals	Manchester City	4.04762
Penalties Made	Arsenal	0.47619
Assists	Manchester City	3.2381

→Sau khi đã có danh sách results gồm các đội dẫn đầu từng chỉ số, chương trình sử dụng Counter để đếm số lần mỗi đội xuất hiện trong danh sách này, tức là số lần một đội có giá trị cao nhất ở một chỉ số nào đó. Kết quả đếm được lưu trong team_frequencies, cung cấp thông tin về tần suất mỗi đội dẫn đầu các chỉ số.Sau đó, nó sử dụng tabulate để in ra bảng tần suất, cho biết số lần mỗi đội xuất hiện ở vị trí dẫn đầu.

➔ Cuối cùng, chương trình xác định đội có phong độ tốt nhất dựa trên đội có tần suất dẫn đầu cao nhất trong frequency_table.

```
# Đếm số lần mỗi đội có giá trị cao nhất ở các chỉ số
team_frequencies = Counter([result[1] for result in results])

# Chuyển kết quả đếm tần suất thành dạng bảng, sắp xếp giảm dần theo số lần
frequency_table = sorted(
    [[team, freq] for team, freq in team_frequencies.items()],
    key=lambda x: x[1],
    reverse=True
)

# In bảng tần suất của từng đội
print("\nTần suất các đội bóng xuất hiện ở vị trí dẫn đầu:")
print(tabulate(frequency_table, headers=["Team", "Số lần"], tablefmt="grid"))

# Xác định đội có phong độ tốt nhất dựa trên tần suất cao nhất
best_team = frequency_table[0][0]
best_team_count = frequency_table[0][1]
print(f"Đội có phong độ tốt nhất là: {best_team} với số lần xuất hiện ở vị trí dẫn đầu: {best_team_count}")
print("=> Đây là đội có phong độ tốt nhất giải Ngoại Hạng Anh mùa 2023-2024")
```

Team	Số lần
Manchester City	54
Liverpool	31
Arsenal	13

```
Đội có tần suất cao nhất ở chỉ số là: Manchester City với số lần là: 54
==> Sẽ là có phong đô tốt nhất giải ngoại Hang Anh mùa 2023-2024
```

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 sử dụng**: pandas, sklearn, numpy, matplotlib

- **pandas**: Thư viện này được dùng để đọc dữ liệu từ file CSV chứa thông tin cầu thủ và thực hiện các thao tác xử lý dữ liệu trước khi áp dụng các thuật toán.

- **StandardScaler từ sklearn.preprocessing**: Sử dụng để chuẩn hóa dữ liệu, đảm bảo rằng mỗi đặc trưng đều có trung bình là 0 và độ lệch chuẩn là 1, giúp cải thiện tính ổn định và hiệu quả của thuật toán K-means.

- **PCA từ sklearn.decomposition**: Dùng để giảm số chiều của dữ liệu, giúp đơn giản hóa và tạo điều kiện thuận lợi cho việc trực quan hóa các cụm cầu thủ.

- **numpy**: Thư viện này hỗ trợ các phép tính toán số học và thao tác trên mảng, giúp thực hiện các bước quan trọng trong K-means như tính khoảng cách Euclidean và trung tâm của các cụm.

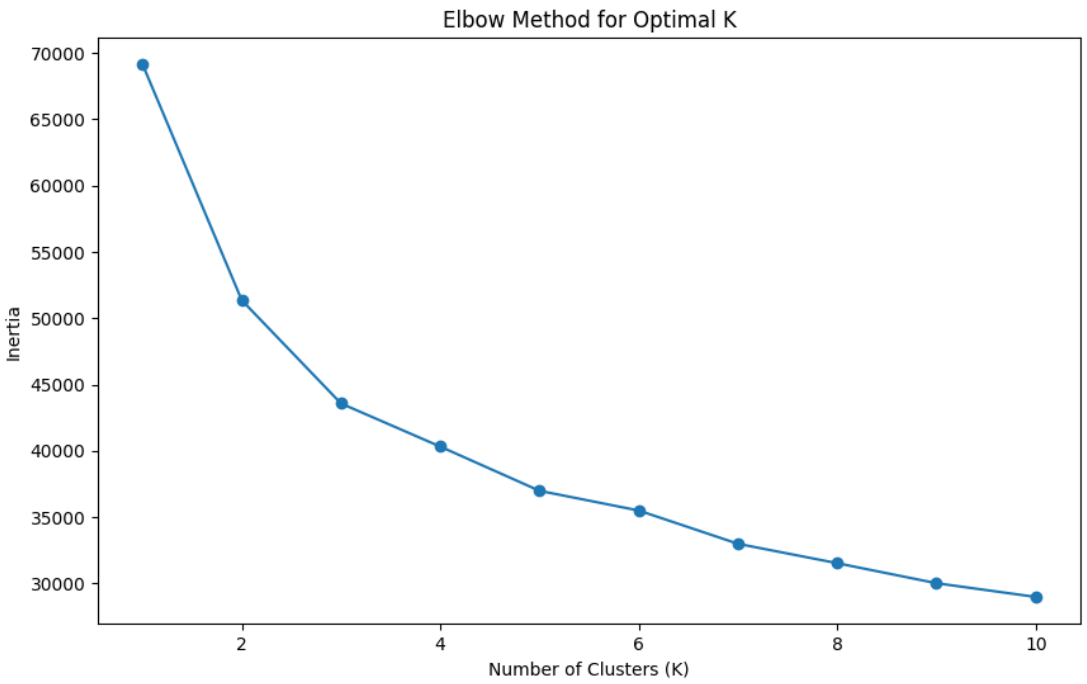
- **matplotlib**: Được sử dụng để tạo các biểu đồ nhằm trực quan hóa quá trình phân cụm và so sánh cầu thủ.

- **argparse**: Thư viện này cho phép chương trình nhận các tham số từ dòng lệnh, cho phép người dùng chỉ định hai cầu thủ muốn so sánh khi chạy chương trình từ terminal.

***Thực hiện:**

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 ta sẽ thu được biểu đồ.



- Dựa vào biểu đồ Silhouette Score trên, ta thấy: nên lấy khoảng k = 4 . Đây là vị trí mà độ giảm của inertia bắt đầu chậm lại, cho thấy rằng việc tăng số cụm lên cao hơn sẽ không giảm nhiều độ biến dạng nữa.
- Tiếp theo, sử dung pandas để đọc file csv chứa dữ liệu và xử lý các dữ liệu trước khi đưa vào chuẩn hoá.

```
data = pd.read_csv('results.csv')
data = data.select_dtypes(exclude=['object'])
data = data.fillna(data.mean())
```

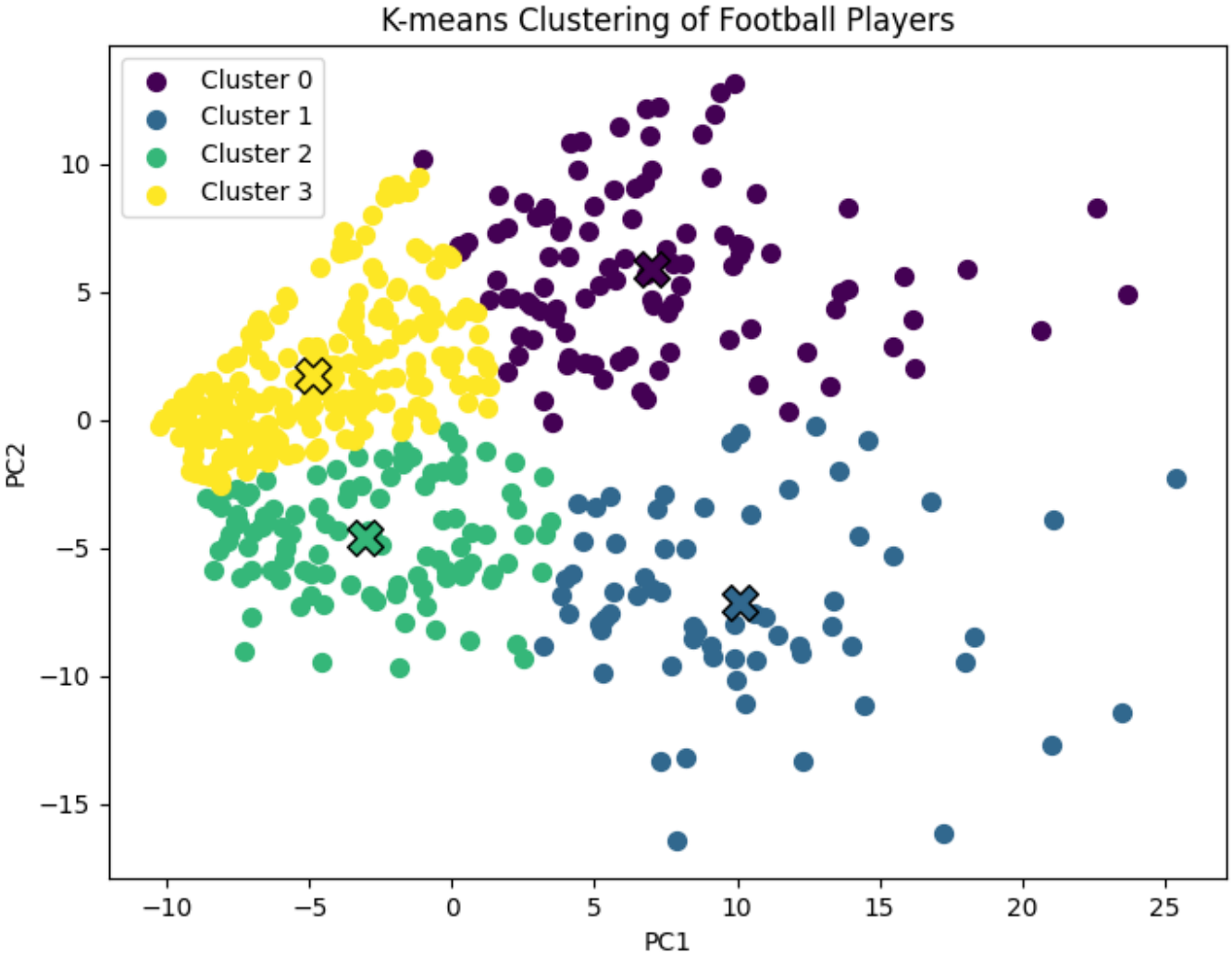
- Chuẩn hoá dữ liệu bằng StandardScaler rồi dùng PCA giảm số chiều xuống

```python
# 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 = 4 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

```python
k = 4

# 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
```

Kết quả :



2.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 để lấy thông số đầu vào.

```
    # Khởi tạo parser để lấy thông số đầu vào
    parser = argparse.ArgumentParser(description='Compare two players using radar chart.')
    parser.add_argument('--p1', type=str, required=True, help='Player 1 name')
    parser.add_argument('--p2', type=str, required=True, help='Player 2 name')
    parser.add_argument('--Attribute', type=str, required=True, help='List of attributes to compare, separated by commas')

    args = parser.parse_args()\
```

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

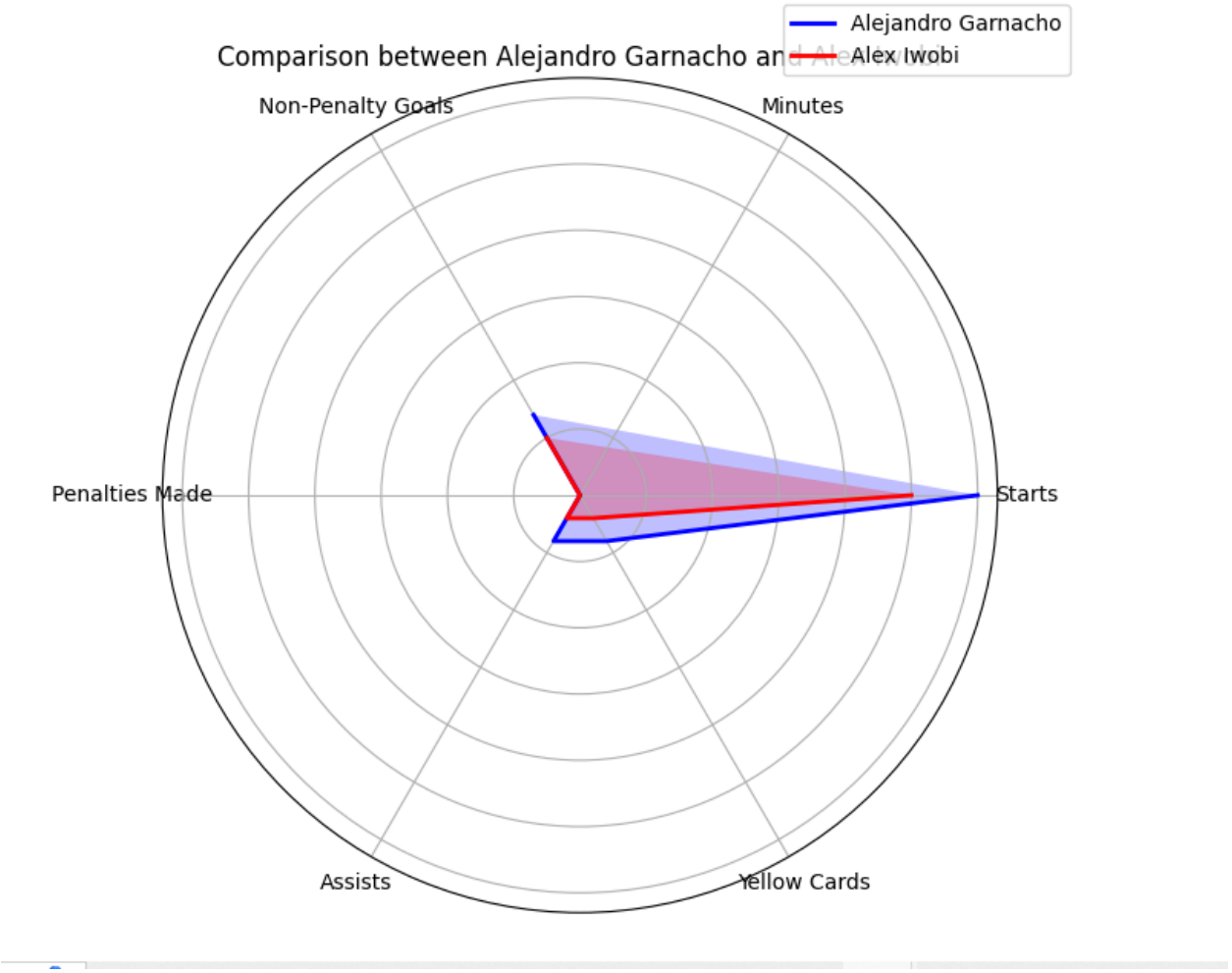
```
# Đọc dữ liệu từ file CSV
data = pd.read_csv('results.csv')
player1 = args.p1
player2 = args.p2
attributes = args.Attribute.split(',')
for attr in attributes:
    data[attr] = pd.to_numeric(data[attr], errors='coerce')

# Vẽ biểu đồ radar
```

- Ở hàm plot_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.

```python
def radar_chart(data, player1, player2, attributes):

    p1_data = data[data['Player Name'] == player1].iloc[0][attributes].values.astype(float)
    p2_data = data[data['Player Name'] == player2].iloc[0][attributes].values.astype(float)

    # Số lượng thuộc tính
    num_vars = len(attributes)

    # Tạo các góc cho biểu đồ radar
    angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

    # Khép kín biểu đồ
    p1_data = np.concatenate((p1_data, [p1_data[0]]))
    p2_data = np.concatenate((p2_data, [p2_data[0]]))
    angles += angles[:1]

    fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(polar=True))

    # Vẽ biểu đồ radar cho từng cầu thủ
    ax.plot(angles, p1_data, color='blue', linewidth = 2, linestyle='solid', label=player1)
    ax.fill(angles, p1_data, color='blue', alpha=0.25)
```

Ví dụ: python bai-3-2.py --p1 "Aaron Cresswell" --p2 "Adam Smith" --Attribute "Age,Matches Played,Starts,Minutes,Non-Penalty Goals,Penalties Made,Assists,Yellow Cards,Red Cards"

Biểu đồ rada sẽ được như sau:



Câu 4: 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.

***Thư viện sử dụng**: BeautifulSoup, requests

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

***Thực hiện: Xem trong code**

- Đề xuất phương pháp định giá cầu thủ: Phương pháp Dựa trên Hiệu Suất (Performance-based Valuation)

- **Dữ liệu Hiệu suất**: Sử dụng các số liệu thống kê như số bàn thắng, số pha kiến tạo, tỷ lệ chuyền chính xác, số lần cản phá (đối với thủ môn), v.v. Các chỉ số này cho thấy giá trị trực tiếp của cầu thủ trong các trận đấu.
- **Xếp hạng & Điểm số**: Sử dụng các công cụ phân tích như Opta, StatsBomb, hoặc Transfermarkt để có cái nhìn tổng quan về hiệu suất của cầu thủ dựa trên các chỉ số tổng hợp.
- **Mô hình Máy học (Machine Learning)**: Xây dựng mô hình máy học để dự đoán giá trị cầu thủ dựa trên các thông số đầu vào như tuổi, vị trí chơi, số phút thi đấu, và số liệu thống kê hiệu suất.