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:Ngô Hoàng Nam

Mã sinh viên: B22DCKH079

Hà Nội, ngày 3 tháng 11 năm 2024

Mục lục

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.

***Ý TƯỞNG:**

- **Thu thập dữ liệu**: Sử dụng requests để truy cập URL của giải Ngoại hạng Anh mùa 2023-2024, lấy HTML, và dùng BeautifulSoup để phân tích và trích xuất các bảng <table> chứa thông tin.

*** 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 (bảng này chứa thông tin về các đội trong giải đó). **(Hình 1.1)**

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

    # Tìm bảng chứa thông tin các đội bóng trong mùa giải 2023-2024
    table = soup.find( name: 'table', attrs: {
        'class': 'stats_table sortable min_width force_mobilize',
        'id': 'results2023-202491_overall'
    })

    # Danh sách chứa dữ liệu đội bóng và URL
    team_data = []
```

Hình 1.1

Tiếp theo, khởi tạo danh sách team_data để lưu thông tin các đội bóng, bao gồm tên đội và URL. Tìm các thẻ <a> trong bảng đã xác định, chỉ xử lý những thẻ có chứa chuỗi "squads" trong href để lấy tên đội và URL, sau đó lưu vào team_data.

Sau đó, tạo danh sách players_data để chứa thông tin toàn bộ cầu thủ trong giải và gọi hàm Crawl_Data_For_Each_Team để thực hiện yêu cầu đề bài.

- **(Hình 1.2)**

⇨ **Đã có danh sách thông tin cần thiết về các đội bóng**

```
if table:
    # Tìm thẻ <tbody> trong <table>
    tbody = table.find('tbody')
    if tbody:
        # Lấy tất cả các thẻ <a> có định dạng như yêu cầu trong <tbody>
        teams = tbody.find_all( name: '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 kqua <tbody>.")
else:
    print("Không tìm thấy thẻ <table>.")

# #  Danh sach chứa từng cầu thủ của đội bóng
players_data = []
players_data = Crawl_Data_For_Each_Team(players_data, team_data)

# Sắp xếp dữ liệu theo first name và tuổi giảm dần
players_data = sorted(players_data, key=lambda x: (x[0].split()[0], -int(x[4])))
```

Hình 1.2

Trong hàm Crawl_Data_For_Each_Team, chúng ta sẽ duyệt qua từng đội trong danh sách team_data và lấy URL của đội đó để truy xuất mã HTML.

- Sau khi lấy được HTML, khởi tạo danh sách player_data_tmp để lưu thông tin cầu thủ của đội và một map mp với khóa là tên cầu thủ và giá trị là danh sách chứa thông tin chỉ số của cầu thủ.
- Tiếp theo, xử lý 10 thẻ <table>, mỗi thẻ chứa thông tin về cầu thủ với các class giống nhau nhưng id khác nhau, giúp dễ dàng tìm kiếm.
- Bắt đầu với bảng Standard Stats có id là “stats_standard_9”. Tìm thẻ <tbody> chứa dữ liệu và sau đó tìm tất cả các thẻ <tr> chứa thông tin cầu thủ. Duyệt qua các thẻ <tr> này, tìm thẻ <td> có “data-stat = minutes” để lấy tổng thời gian thi đấu. Nếu thời gian thi đấu nhỏ hơn 90 phút, bỏ qua cầu thủ đó; nếu không, gọi một hàm con để xử lý thông tin và nhận về một danh sách.

Trong hàm Data_Processing_of_Footballer, chúng ta sẽ thu thập thông tin cho từng chỉ số thông qua các thẻ <td> với thuộc tính “data-stat” riêng biệt. Hàm này sẽ trả về một danh sách chứa thông tin các chỉ số của cầu thủ. Đồng thời, chúng ta sẽ tạo một cặp key-value để lưu trữ trong map, trong đó key là tên chỉ số và value là giá trị tương ứng của chỉ số đó.

```
# Hàm xử lý dữ liệu cầu thủ
def Data_Processing_of_Footballer(tmp_tr, team_name, player_data_tmp, mp):  1 usage
    # Lấy thông tin cầu thủ
    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 = extract_data(tmp_tr,  stat: 'position')
    player_age = extract_data(tmp_tr,  stat: 'age')

    # Playing time
    player_games = extract_data(tmp_tr,  stat: 'games')
    player_games_starts = extract_data(tmp_tr,  stat: 'games_starts')
    player_minutes = extract_data(tmp_tr,  stat: 'minutes')

    # Performance
    player_goals_pens = extract_data(tmp_tr,  stat: 'goals_pens')
    player_pens_made = extract_data(tmp_tr,  stat: 'pens_made')
    player_assists = extract_data(tmp_tr,  stat: 'assists')
    player_cards_yellow = extract_data(tmp_tr,  stat: 'cards_yellow')
    player_cards_red = extract_data(tmp_tr,  stat: 'cards_red')

    # Expected
    player_xg = extract_data(tmp_tr,  stat: 'xg')
    player_npxg = extract_data(tmp_tr,  stat: 'npxg')
    player_xg_assist = extract_data(tmp_tr,  stat: 'xg_assist')
```

```
    # Thêm thông tin cầu thủ vào danh sách
    tmp = [
            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_stats_per90

    player_data_tmp.append(tmp)
    mp[player_name] = player_data_tmp[-1]

    return player_data_tmp
```

Tiếp theo, chúng ta sẽ xử lý bảng có id là “stats_keeper_10”. Đầu tiên, tìm thẻ <tbody> và các thẻ <tr> tương tự như ở bảng trước. Tạo một danh sách tạm thời tên là list_tmp để chứa tên các thủ môn.

- Duyệt qua từng thẻ <tr>, lấy tên shooting từ thẻ <th> và kiểm tra xem thủ môn này có tồn tại trong danh sách player_data_tmp không.
- Nếu có, gọi hàm để xử lý thông tin của shooting đó (hàm này cũng sẽ trả về một danh sách).
- Để kết hợp thông tin của shooting với thông tin từ bảng trước, sử dụng phép nối danh sách thông qua toán tử + và thêm tên thủ môn vào list_tmp.

- **(Hình 1.7)**

```
# Hàm xử lý dữ liệu Shooting
def Data_Processing_of_Shooting(player):  1 usage
    # Helper function to extract data with a default value
    def extract_data(stat):
        return player.find('td', {'data-stat': stat}).get_text(strip=True) if player.find('td', {'data-stat': stat}) else "N/a"

    # Extracting shooting statistics using the helper function
    player_Gls = extract_data('goals')
    player_Sh = extract_data('shots')
    player_SoT = extract_data('shots_on_target')
    player_SoTP = extract_data('shots_on_target_pct')
    player_Sh90 = extract_data('shots_per90')
    player_Sot90 = extract_data('shots_on_target_per90')
    player_GSh = extract_data('goals_per_shot')
    player_GSoT = extract_data('goals_per_shot_on_target')
    player_Dist = extract_data('average_shot_distance')
    player_FK = extract_data('shots_free_kicks')
    player_PK = extract_data('pens_made')
    player_PKatt = extract_data('pens_att')
    player_xG = extract_data('xg')
    player_npxG = extract_data('npxg')
    player_npxGSh = extract_data('npxg_per_shot')
    player_GxG = extract_data('xg_net')
    player_npGxG = extract_data('npxg_net')

    # Trả về list chỉ số shooting
    return [
        player_Gls, player_Sh, player_SoT, player_SoTP, player_Sh90,
        player_Sot90, player_GSh, player_GSoT, player_Dist,
```

- 8 bảng còn lại sẽ xử lý giống với bảng trên , mỗi bảng có một hàm xử lý thông tin riêng biệt (xem code để rõ ràng hơn).
- Sau khi xử lý xong 10 bảng, ta sẽ thm list ***player_data_tmp*** vào ***players_data*** và sẽ tạm dừng khoảng 1.0 giây để xử lý đội tiếp theo (tránh bị web chặn).

Sau khi đã xử lý và thu thập thông tin các cầu thủ của các đội, chúng ta sẽ thực hiện các bước sau:

- Sắp xếp danh sách players_data theo thứ tự từ điển của first name và theo tuổi giảm dần (nếu first name bằng nhau).
- Sử dụng thư viện pandas để tạo một DataFrame từ players_data.
- Thêm tên cho các cột trong DataFrame để định danh các chỉ số và thông tin.
- Cuối cùng, lưu DataFrame vào file "results.csv" thông qua hàm của thư viện pandas.

- . **(Hình 1.10)**

```
# # 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",
                                   "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", "GGoalkeeping_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",
```

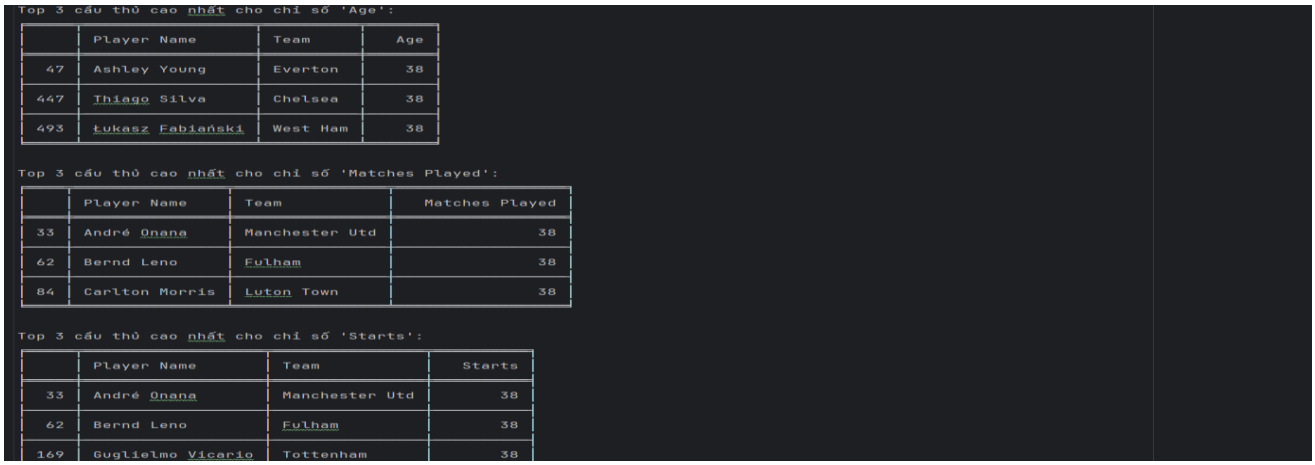
CÂU 2: Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số. Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉ số cho các cầu thủ trong toàn giải và của mỗi đội. Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội. Tìm đội bóng có chỉ số điểm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

***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 hơvà 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

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(3, column)*** 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. Tương tự với 3 cầu thủ có chỉ số thấp nhất. **(Hình 2.4)**

Hình 2.3



```
Top 3 cầu thủ cao nhất cho chỉ số 'Age':
╒═══════╤═══════════════════╤═══════════╤═══════╕
│       │ Player Name       │ Team      │   Age │
╞═══════╪═══════════════════╪═══════════╪═══════╡
│    47 │ Ashley Young      │ Everton   │    38 │
├───────┼───────────────────┼───────────┼───────┤
│   447 │ Thiago Silva      │ Chelsea   │    38 │
├───────┼───────────────────┼───────────┼───────┤
│   493 │ Łukasz Fabiański  │ West Ham  │    38 │
╘═══════╧═══════════════════╧═══════════╧═══════╛

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 │
╘═══════╧═══════════════════╧═══════════════╧═══════════════╛

Top 3 cầu thủ cao nhất cho chỉ số 'Starts':
╒═══════╤═══════════════════╤═══════════════╤═══════════╕
│       │ Player Name       │ Team          │    Starts │
╞═══════╪═══════════════════╪═══════════════╪═══════════╡
│    33 │ André Onana       │ Manchester Utd│        38 │
├───────┼───────────────────┼───────────────┼───────────┤
│    62 │ Bernd Leno        │ Fulham        │        38 │
├───────┼───────────────────┼───────────────┼───────────┤
│   169 │ Guglielmo Vicario │ Tottenham     │        38 │
```

```python
def get_top3(df, column_to_analyze):  1 usage
    # Ghi kết quả tìm kiếm Top3 cao nhất vào file Top3NguoiChiSoCaoNhat.txt
    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 tìm kiếm Top3 cao nhất vào file Top3NguoiChiSoCaoNhat.txt>>>>>>")

    # Ghi kết quả tìm kiếm Top3 thấp nhất vào file Top3NguoiChiSoThapNhat.txt
    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 tìm kiếm Top3 thấp nhất vào file Top3NguoiChiSoThapNhat.txt>>>>>>")
```

Hình 2.4

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.**
   - Xử lý trong hàm **get_statistics**.

- Tính trung vị bằng gọi hàm ***median***, trung bình gọi hàm ***mean***, độ lệch chuẩn gọi hàm ***std***, làm tròn gọi hàm ***round***.
- Tạo một ***DataFrame*** là ***overall_df*** để lưu trữ các giá trị thông kê của toàn giải sau khi đã dùng 4 hàm ở trên

```python
def get_statistics(df, columns_to_analyze):  1 usage
    # Calculate median, mean, and standard deviation for overall
    median_all = df[columns_to_analyze].median().round(2)
    mean_all = df[columns_to_analyze].mean().round(2)
    std_all = df[columns_to_analyze].std().round(2)

    # Create a DataFrame to store the overall statistics
    overall_data = {
        'STT': [0],
        'Team': ['all']
    }

    for col in columns_to_analyze:
        overall_data[f'Median of {col}'] = [median_all[col]]
        overall_data[f'Mean of {col}'] = [mean_all[col]]
        overall_data[f'Std of {col}'] = [std_all[col]]

    overall_df = pd.DataFrame(overall_data)

    # Calculate median, mean, and standard deviation for each team
    median_team = df.groupby('Team')[columns_to_analyze].median().round(2)
    mean_team = df.groupby('Team')[columns_to_analyze].mean().round(2)
    std_team = df.groupby('Team')[columns_to_analyze].std().round(2)

    # Create a DataFrame to store the team statistics
    team_data = {
        'STT': range(1, len(median_team) + 1),
        'Team': median_team.index
    }
```

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 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). Xem code để rõ hơn về cái này.
   - Với toàn giải, thì sử dụng thư viện **matplotlib** với **seaborn** để vẽ với các thông số biểu đồ đã được tạo trong code

   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.

```
    os.makedirs(output_folder_1)

# 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( visible: 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")

# 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:
```

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.**
    - Xử lý 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**.(Hình 2.10)**

```python
def get_best_team(df, columns_to_analyze):

    columns_to_analyze = df.columns[8:]  # Chỉ chọn các cột chỉ số từ cột "Non-Penalty Goals" trở đi
    df[columns_to_analyze] = df[columns_to_analyze].apply(pd.to_numeric, errors='coerce')

  # Nhóm theo 'Tên Đội' và tính trung bình các chỉ số
    team_summary = df.groupby('Team')[columns_to_analyze].mean()
```

Hình 2.10

- Tạo một list là ***results*** để chứa kết quả sau khi xử lý tìm đội có giá trị cao nhất ở các chỉ số **(Hình 2.11 và 2.12).** Sau đó, đếm tần suất đội đó có tên trong ***results***. Kết quả dự đoán sẽ dựa trên đội có tần suất cao nhất chứng tỏ đội đó có hiệu suất, phong độ tốt nhất.**(Hình 2.13 và 2.14)**

```python
# Tạo một danh sách để chứa kết quả
results = []

# Tìm đội có giá trị cao nhất ở từng chỉ số
for column in columns_to_analyze:
    best_team = team_summary[column].idxmax()
    max_value = team_summary[column].max()
    results.append([column, best_team, max_value])

# In kết quả dưới dạng bảng
headers = ["Chỉ số", "Team", "Giá trị"]
print(tabulate(results, headers=headers, tablefmt="grid"))
```

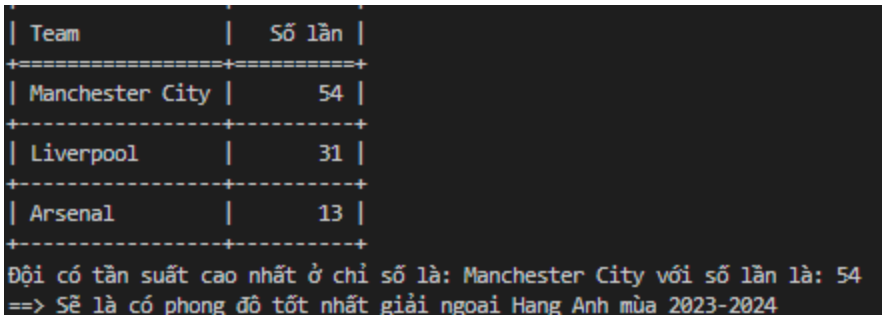
Hình 2.11

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

Hình 2.12

```python
# Đếm tần suất của từng đội
team_counts = Counter([row[1] for row in results])

# Chuyển kết quả đếm tần suất thành dạng bảng
frequency_table = [[team, count] for team, count in team_counts.items()]
frequency_table.sort(key=lambda x: x[1], reverse=True)

# In bảng tần suất của từng đội
print("\nTần suất của từng đội bóng:")
print(tabulate(frequency_table, headers=["Team", "Số lần"], tablefmt="grid"))

print("Đội có tần suất cao nhất ở chỉ số là: " + str(frequency_table[0][0]) + " với số lần là: " + str(frequency_table[0][1]))
print("==> Sẽ là có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024")
```

Hình 2.13



Hình 2.14

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 Silhouette Score** .
- Dùng thư viện **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 Silhouette Score** (code chạy trong file **Silhouette Score.py**) ta sẽ thu được biểu đồ. **(Hình 3.1)**

Hình 3.1

- Dựa vào biểu đồ **Silhouette Score** trên, ta thấy:
  - **K = 2** có **Silhouette Score** cao nhất (khoảng 0.30), cho thấy đây là số cụm có chất lượng phân cụm tốt nhất vì các điểm dữ liệu được phân cách rõ ràng giữa các cụm.
  - **K = 3** có **Silhouette Score** thấp hơn một chút (khoảng 0.28), nhưng vẫn tương đối cao, cho thấy cũng là lựa chọn khả thi.
  - Từ **K = 4** trở lên, **Silhouette Score** giảm đáng kể, chỉ dao động từ khoảng 0.16 đến 0.22, cho thấy chất lượng phân cụm không còn tốt nữa và các cụm không được phân biệt rõ ràng.

⇨ Có thể chọn ***K = 2 hoặc K = 3*** vì phù hợp với nội dung của phương pháp.

- Tiếp theo, sử dung **pandas** để đọc file csv chứa dữ liệu và xử lý các dữ liệu trước khi đưa vào chuẩn hoá. **(Hình 3.2)**

```
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())
```

Hình 3.2

- Chuẩn hoá dữ liệu bằng **StandardScaler** rồi dùng **PCA** giảm số chiều xuống 2. **(Hình 3.3)**

```
# 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'])
```

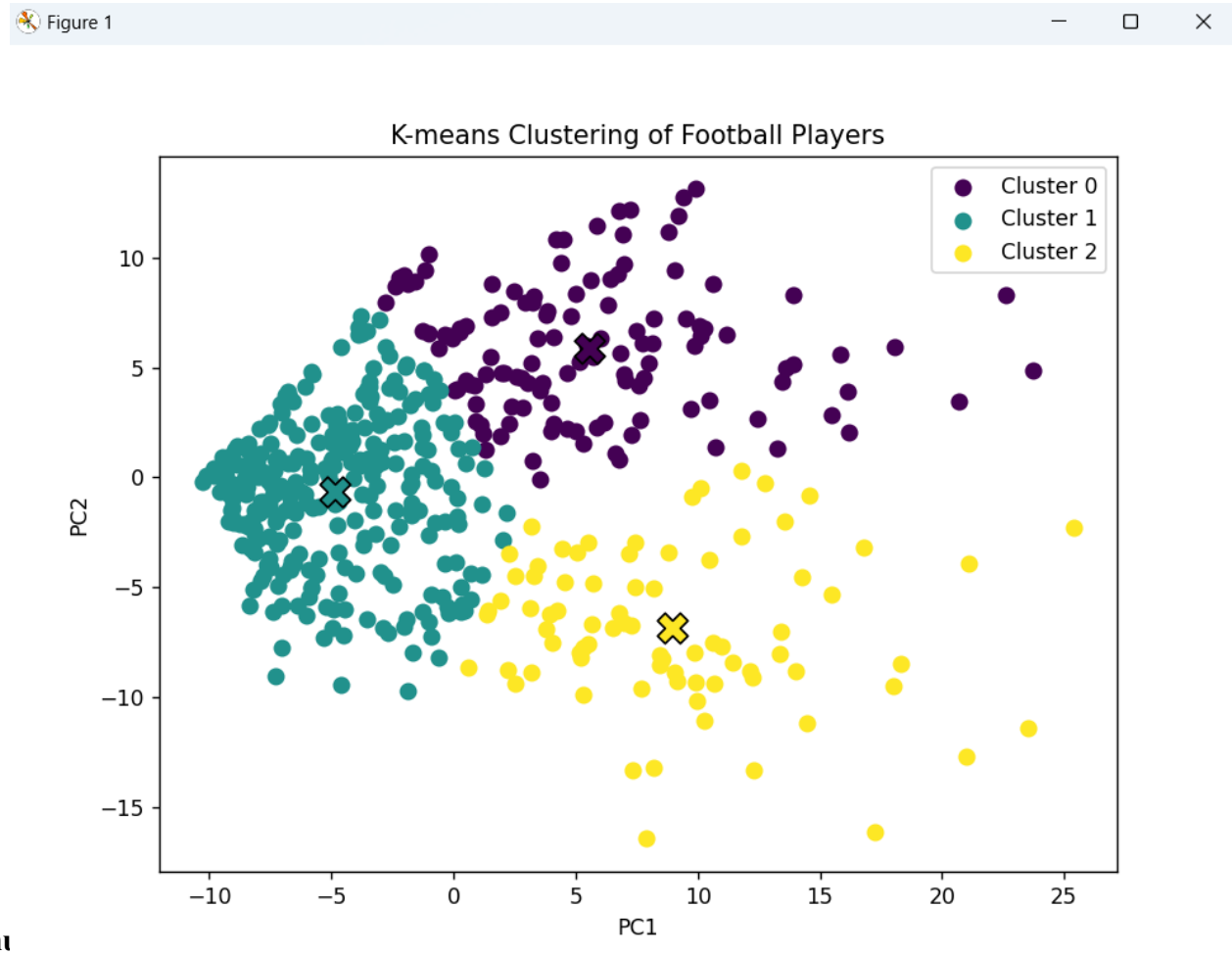
Hình 3.3

- Sau đó dùng ***K-means*** để phân loại cầu thủ với số lượng cụm (nhóm) ***K = 3*** đã suy ra từ **phương pháp Silhouette Score** 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 **Cau3-1.py**). **(Hình 3.4)**

```
# 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
```

Hình 3.4

- **Kết quả:**



2. **Viết chu**
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. **(Hình 3.5)**

```
def main():
    # 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, separate
```

Hình 3.5

- 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**)*. **(Hình 3.6)**

```
# Đọc dữ liệu từ file CSV
data = pd.read_csv('results.csv')

player1 = args.p1
player2 = args.p2
attributes = args.Attribute.split(',')

# Chuyển các cột dữ liệu về dạng số
for attr in attributes:
    data[attr] = pd.to_numeric(data[attr], errors='coerce')

# Vẽ biểu đồ radar
plot_radar_chart(data, player1, player2, attributes)
```

Hình 3.6

- 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*. **(Hình 3.7)**

```
def plot_radar_chart(data, player1, player2, attributes):
    # Lấy dữ liệu của hai cầu thủ
    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()
```

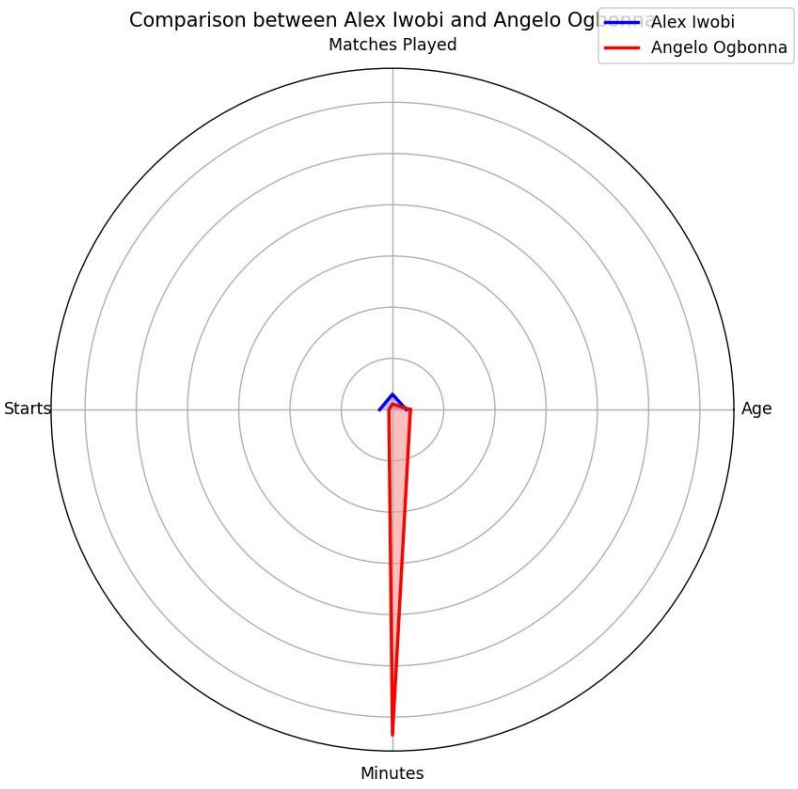
Hình 3.7

- Thiết lập chỉ số, thuộc tính để vẽ biểu đồ rada (xem chi tiết hơn trong file **Cau3-2.py**).

 Cách chạy file Cau3-2.py để so sánh 2 cầu thủ: save file trước -> vào terminal gõ theo format này: *python Cau3-2.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 Cau3-2.py --p1 "Alex Iwobi" --p2 "Angelo Ogbonna" --Attribute "Age,Matches Played,Starts,Minutes"*

⇨ **Biểu đồ rada sẽ được như sau: (Hình 3.8)**



Hình 3.8

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 cần cài:** pandas, BeautifulSoup, requests.

 Cách cài: vào CMD gõ *"pip install pandas BeautifulSoup requests"* và ấn ENTER.

***Ý TƯỞNG:**

- Dùng **requests** để lấy HTML từ URL của trang web. Sau đó, dùng **BeautifulSoup** phân tích và xử lý thẻ <*table*>.
- Dùng **pandas** để lưu dữ liệu xử lý được vào file csv.

***TRONG CODE:**

1. **Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024 từ trang web17 https://www.footballtransfers.com.**

- Lấy HTML từ trang web thông qua URL và dùng **BeautifulSoup** để tìm thẻ <*table*> đầu tiên là nơi chứa thông tin các đội bóng. Sau đó, tạo list ***teams_data*** lưu thông tin các đội bóng và tìm thẻ <*tbody*> trong thẻ <*table*> vừa tìm được và tiếp tục tìm các thẻ <*a*> trong thẻ
<*tbody*>. Duyệt lần lượt các thẻ <*a*> và lấy nội dung của thẻ <*a*> và nội dung của *href* trong

thẻ <*a*> đang xét này chính là tên đội bóng và URL của đội đó => lưu vào trong ***teams_data.*** **(Hình 4.1)**

```
if __name__ == "__main__":
    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')
```

Hình 4.1

- Duyệt từng team trong ***teams_data***, lấy HTML của đội từ URL của đội. Tiếp là tìm thẻ <*table*> đầu tiên có *"class : table table-striped-rowspan ft-table mb-0"* và tìm thẻ <*tbody*> từ thẻ <*table*>. Tiếp tục, tìm tất cả thẻ <*tr*> trong thẻ <*tbody*>, duyệt lần lượt các thẻ <*tr*> và chỉ xử lý các thẻ có nội dung *class* là *"odd"* hoặc *"even"* và từ đó lấy tên cầu thủ và giá chuyển nhượng cầu thủ đó => lưu vào list ***players_data*** và lưu vào file csv. **(Hình 4.2)**

```
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)

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

Hình 4.2