HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BÁO CÁO BÀI TẬP
MÔN HỌC: LẬP TRÌNH VỚI PYTHON

Giảng viên : Thầy Kim Ngọc Bách

Sinh viên : Bùi Đức Mạnh

Lớp : D22CQCN03-B

Mã sinh viên : B22DCCN519

Hà Nội, năm 2024

I. Viết chương trình Python thu thập dữ liệu phân tích cầu thủ

1. Mục tiêu

Mã nguồn này có mục đích lấy và xử lý dữ liệu của các cầu thủ từ trang web FBref, cu thể là 1 từ mùa giải 2023-2024 tại Giải Ngoại hạng Anh (Premier League). Dữ liệu thu thập bao gồm các thông tin cơ bản về cầu thủ như số trận đã chơi, số phút thi đấu, các thông số thống kê về hiệu suất ghi bàn, kiến tạo, số thẻ phạt, và một số chỉ số nâng cao khác.

2. Công cụ và thư viện

```
1  from selenium import webdriver
2  from selenium.webdriver.common.by import By
3  import time
4
5  from player import Player
6  from player import Player_Manager
7  from player import Squad
8  from player import Squad_Manager
9
10 player_manager = Player_Manager()
11 squad_manager = Squad_Manager()
```

- selenium: Thư viện dùng để tự động tương tác với trình duyệt web.
- selenium.webdriver.common.by: Cung cấp các phương thức để định vị các thành phần trên trang web.
- time: Thư viện cung cấp các hàm xử lý thời gian.
- player: Module chứa các định nghĩa liên quan đến cầu thủ, bao gồm class Player và Player_Manager.
- player_manager: Class quản lý danh sách các cầu thủ.
- squad: Module chứa các định nghĩa liên quan đến đội bóng, bao gồm class Squad và Squad_Manager.
- squad_manager: Class quản lý danh sách các đội bóng.
- tieu_de: Module chứa các hàm liên quan đến tiêu đề và định dạng dữ liệu xuất ra file CSV.
- csv: Thư viện dùng để xử lý file định dạng CSV.

3. Phân tích sơ lược code

3.1 Hàm

3.1.1 validdata(n)

Hàm này kiểm tra giá trị đầu vào n. Nếu n là rỗng, hàm trả về "N/a" (Không có dữ liệu). Ngược lại, hàm chuyển đổi chuỗi n thành giá trị số thực.

3.1.2 GetDataFromWeb(url, Xpath_player, Xpath_squad, Data_Name)

- Đây là hàm chính thực hiện việc lấy dữ liệu từ một trang web. Hàm nhận các tham số sau:

  - url: Địa chỉ URL của trang web chứa dữ liệu.
  - Xpath_player: Biểu thức XPath để định vị bảng dữ liệu cầu thủ.
  - Xpath_squad: Biểu thức XPath để định vị bảng dữ liệu đội bóng.
  - Data_Name: Tên của loại dữ liệu đang được lấy (chủ yếu dùng để đặt tên cho các file lưu trữ).

- Các bước thực hiện của hàm:

  1. Tạo một phiên bản trình duyệt Chrome.
  2. Mở trang web theo địa chỉ URL được cung cấp.
  3. Chờ đợi trong 10 giây (có thể điều chỉnh thời gian chờ).
  4. Tìm bảng dữ liệu cầu thủ theo biểu thức XPath.
  5. Lặp qua từng dòng dữ liệu trong bảng (bỏ qua dòng tiêu đề):
     - Lấy dữ liệu từ từng ô trong dòng.
     - Loại bỏ khoảng trắng thừa, xử lý dấu phẩy (nếu có) và chuyển đổi sang số thực (nếu cần thiết).
     - Bỏ qua một số cột nhất định (ví dụ: năm sinh).
     - Thêm dữ liệu đã xử lý vào một danh sách.
  6. Thực hiện các bước tương tự (4 và 5) để lấy dữ liệu bảng đội bóng.
  7. Đóng trình duyệt.
  8. In ra thông báo hoàn thành việc lấy dữ liệu từ trang web.
  9. Trả về hai danh sách: resultPlayerData chứa dữ liệu cầu thủ và resultSquadData chứa dữ liệu đội bóng.

3.2 Script chính

Script chính thực hiện theo các bước sau:

3.2.1 Khởi tạo các đối tượng quản lý dữ

3.2.2 Lấy dữ liệu từ một trang web

- **Định vị các phần tử:**
  - Sử dụng XPath để xác định chính xác vị trí của bảng dữ liệu cầu thủ và đội bóng trên trang web. XPath là một ngôn ngữ truy vấn để chọn các phần tử trong một tài liệu XML hoặc HTML.

- time.sleep(10): Hàm này tạm dừng chương trình trong 10 giây để đảm bảo trang web được tải đầy đủ trước khi bắt đầu lấy dữ liệu. Tuy nhiên, việc sử dụng time.sleep như vậy không hiệu quả và nên thay thế bằng các cơ chế chờ đợi thông minh hơn dựa trên sự kiện (ví dụ: chờ cho một phần tử cụ thể xuất hiện trên trang).

- **Xử lý dữ liệu:**
  - Lặp qua từng hàng trong bảng dữ liệu:
    - Lấy dữ liệu từ từng ô trong hàng.
    - Làm sạch dữ liệu: Loại bỏ khoảng trắng thừa, chuyển đổi dữ liệu sang định dạng số nếu cần.
    - Bỏ qua các cột không cần thiết.
    - Lưu trữ dữ liệu vào danh sách.
  - Kết quả thu được là hai danh sách: một danh sách chứa dữ liệu của các cầu thủ và một danh sách chứa dữ liệu của các đội bóng.

3.2.3 Xử lý dữ liệu cầu thủ

- **Tìm kiếm cầu thủ:**
  - Kiểm tra xem cầu thủ đã tồn tại trong danh sách các cầu thủ đã được lấy trước đó hay chưa.
  - Nếu chưa tồn tại, tạo một đối tượng Player mới và gán các giá trị cho các thuộc tính của đối tượng này (tên, vị trí, đội bóng, ...).
  - Gán các giá trị thống kê cho cầu thủ (thời gian thi đấu, hiệu suất, chỉ số dự đoán, ...).
  - Thêm cầu thủ vào danh sách các cầu thủ.

3.2.4 Xử lý dữ liệu đội bóng

- **Tìm kiếm đội bóng:**
  - Kiểm tra xem đội bóng đã tồn tại trong danh sách các đội bóng đã được lấy trước đó hay chưa.
  - Nếu chưa tồn tại, tạo một đối tượng Squad mới và gán các giá trị cho các thuộc tính của đối tượng này (tên đội, thống kê đội).
  - Thêm đội bóng vào danh sách các đội bóng.

3.2.5 Lưu trữ dữ liệu vào file CSV

- **Định nghĩa cấu trúc dữ liệu:**

  - Sử dụng module tieu_de để định nghĩa cấu trúc của các dòng dữ liệu trong file CSV (các tiêu đề cột).

- **Viết dữ liệu vào file:**

  - Mở các file CSV để ghi dữ liệu.
  - Lặp qua danh sách các cầu thủ và đội bóng.
  - Chuyển đổi dữ liệu của mỗi đối tượng thành một dòng trong file CSV.
  - Ghi dữ liệu vào file.

II.

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

- Lấy dữ liệu từ file player_data.txt

```
def read_player_data_from_txt(file_path):
    players = []
    with open(file_path, 'r', encoding='utf-8') as file:
        # Bỏ qua dòng tiêu đề
        header = file.readline().strip().split("\t")

        # Đọc từng dòng dữ liệu
        for line in file:
            # Tách các giá trị bằng dấu tab và thêm vào danh sách
            values = line.strip().split("\t")
            players.append(values)
    return players

# Sử dụng hàm để đọc dữ liệu cầu thủ từ file player_data.txt
player_data = read_player_data_from_txt('D:/Python/BTL/bai1/file/player_data.txt')
```

- Với mỗi chỉ số chúng ta sort để xắp xếp thứ tự và top 3 cầu thủ có điểm cao nhất và thấp nhất vào class max_min:

```
list_mm = max_min()

for index, value in enumerate(header):
    if index < 3: continue
    player_data = sorted(player_data, key=lambda x: x[index])
    list_mm.add_max_min([value, player_data[0][0], player_data[1][0], player_data[2][0],
                         player_data[-3][0],player_data[-2][0], player_data[-1][0]])
```

- Lưu dữ liệu vào file result.xlsx

```python
import openpyxl
# Lưu dữ liệu vào file Excel cho player_manager
wb = openpyxl.Workbook()
ws = wb.active
ws.title = "Players"

# Ghi tiêu đề (header)
ws.append(header_max_min)
# Ghi dữ liệu của các cầu thủ
for player in list_mm.list_max_min:
    ws.append(player)

# Lưu vào file Excel
wb.save('D:/Python/BTL/bai2/file/result.xlsx')
print("Exam 1 Success - Player Data Saved")
```

2. Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉsố cho các cầu thủ trong toàn giải và của mỗi đội. Ghi kết quả ra file results2.csv

- Tải dữ liệu và tạo hàm để thêm dữ liệu vào cho mỗi chỉ số

```python
# Tải dữ liệu từ file CSV vào DataFrame df
df = pd.read_csv('D:/Python/BTL/bai2/file/result.xlsx')

def add_statistics_for_team(team_name, numeric_df, table):
    table['Team'].append(team_name)
    for att in numeric_df.columns:
        table['Median of ' + att].append(float(numeric_df[att].median()))
        table['Mean of ' + att].append(float(numeric_df[att].mean()))
        table['Std of ' + att].append(float(numeric_df[att].std()))

numeric_df = df.select_dtypes(include=['float', 'int'])
# Tạo dictionary để lưu kết quả
table = {'Team': []}
```

- Tính toán dữ liệu cho từng cầu thủ trong giải:

```
numeric_df = df.select_dtypes(include=['float', 'int'])
# Tạo dictionary để lưu kết quả
table = {'Team': []}
# Khởi tạo các cột cho Median, Mean, Std của từng thuộc tính số
for att in numeric_df.columns:
    table['Median of ' + att] = []
    table['Mean of ' + att] = []
    table['Std of ' + att] = []
    # Tính toán cho tất cả các đội (team 'all')
add_statistics_for_team('all', numeric_df, table)
teams=['all']
teams.extend(df['team'].unique())
```

- Tính toán dữ liệu cho từng đội trong giải:

```
    # Tính toán cho từng đội
for team in teams[1:]:
        # Bỏ qua 'all' vì đã tính toán trước
    filtered_df = df[df['team'] == team]
    numeric_df = filtered_df.select_dtypes(include=['float', 'int'])
    add_statistics_for_team(team, numeric_df, table)
  # Tạo DataFrame từ dictionary 'table'
result2 = pd.DataFrame(table)
```

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

- Lấy dữ liệu cần thiết và tạo các thư mục để lưu cái file .png

```
# Đọc dữ liệu
df = pd.read_csv('D:/Python/BTL/bai2/file/result.xlsx')

# Tạo thư mục để lưu biểu đồ nếu chưa tồn tại
output_folder = 'D:/Python/BTL/bai2/file/output_histograms'
os.makedirs(output_folder, exist_ok=True)
```

- Vẽ historgram cho các cầu thủ trong giải đấu

```
numeric_df = df.select_dtypes(include=['float', 'int'])
num_attributes = len(numeric_df.columns)
num_cols = 4
num_rows = math.ceil(num_attributes / num_cols)

plt.figure(figsize=(16, 10))  # Tăng chiều cao
for idx, att in enumerate(numeric_df.columns, 1):
    plt.subplot(num_rows, num_cols, idx)
    plt.hist(numeric_df[att], bins=10, alpha=0.7, color='blue', edgecolor='black')
    plt.title(f'Histogram of {att} (All Teams)', fontsize=10)  # Giảm kích thước font tiêu đề
    plt.xlabel(att, fontsize=8)  # Giảm kích thước font nhãn
    plt.ylabel('Frequency', fontsize=8)  # Giảm kích thước font nhãn
    plt.grid(axis='y', alpha=0.75)

plt.subplots_adjust(hspace=0.5, wspace=0.3)  # Điều chỉnh khoảng cách giữa các ô con
plt.savefig(os.path.join(output_folder, 'all_teams_histogram.png'))
plt.close()
```

- Vẽ historgram cho các cầu thủ trong mỗi đội

```
teams = df['team'].unique()
for
    (variable) num_attributes_team: int
    t_dtypes(include=['float', 'int'])
    num_attributes_team = len(team_df.columns)
    num_rows_team = math.ceil(num_attributes_team / num_cols)

    plt.figure(figsize=(16, 10))  # Tăng chiều cao
    for idx, att in enumerate(team_df.columns, 1):
        plt.subplot(num_rows_team, num_cols, idx)
        plt.hist(team_df[att], bins=10, alpha=0.7, color='green', edgecolor='black')
        plt.title(f'Histogram of {att} ({team})', fontsize=10)  # Giảm kích thước font tiêu đề
        plt.xlabel(att, fontsize=8)  # Giảm kích thước font nhãn
        plt.ylabel('Frequency', fontsize=8)  # Giảm kích thước font nhãn
        plt.grid(axis='y', alpha=0.75)

    plt.subplots_adjust(hspace=0.5, wspace=0.3)  # Điều chỉnh khoảng cách giữa các ô con
    plt.savefig(os.path.join(output_folder, f'{team}_histogram.png'))
    plt.close()
```

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

- Chuẩn bị dữ liệu và lọc thêm tên cái đội bóng để tìm kiếm chỉ số có giá trị lớn nhất

```
df2 = pd.read_csv('D:/Python/BTL/bai2/file/result2.csv')
df2.drop(index=0, inplace=True)
numeric_columns = df2.select_dtypes(include=['float', 'int']).columns

# Tạo danh sách để lưu kết quả cho mỗi thuộc tính
highest_team_per_stat = []

for column in numeric_columns:
    # Tìm chỉ số dòng có giá trị lớn nhất cho thuộc tính
    max_idx = df2[column].idxmax()

    # Lấy tên đội bóng và giá trị của thuộc tính tại chỉ số này
    max_team = df2.loc[max_idx, 'Team']
    max_score = df2.loc[max_idx, column]

    # Thêm kết quả vào danh sách
    highest_team_per_stat.append({
        'Attribute': column,
        'Team': max_team,
        'Highest Score': max_score
    })
```

- Lưu lại dữ liệu của bảng highest_team_per_stat vào result3.csv

```
# Chuyển đổi danh sách kết quả thành DataFrame
highest_team_per_stat_df = pd.DataFrame(highest_team_per_stat)

    # lưu vào result3.csv
highest_team_per_stat_df.to_csv('D:/Python/BTL/bai2/file/result3.csv', index=False)
```

- In ra mỗi đội bóng có bao nhiêu chỉ số cao nhất theo thứ tự giảm dần

```
# đếm  xem mỗi đội bóng có bao nhiêu chỉ số điểm cao nhất
print(highest_team_per_stat_df['Team'].value_counts())
```

- Theo dữ liệu thu nhập được thì Manchester City đang có phong độ cao nhất với 107 chỉ số đứng đầu

III.

1. Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau

- Xử lý các giá trị thiếu nếu có

```
# Tiền xử lý: Xử lý các giá trị thiếu nếu có
numeric_columns = df.select_dtypes(include=[np.number]).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())
```

- Chuẩn hóa dữ liệu số

```
# Chuẩn hóa dữ liệu số
numerical_data = df.select_dtypes(include=[float, int])
scaler = StandardScaler()
X = scaler.fit_transform(numerical_data)
```

- Tìm số lượng cụm tối ưu

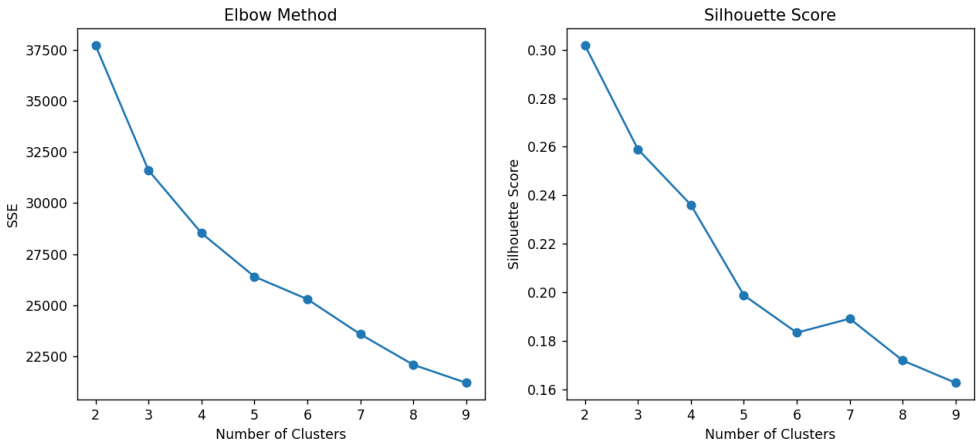
```
# Tìm số lượng cụm tối ưu
sse = []
silhouette_scores = []
for k in range(2, 10):

    # Xử lý giá trị NaN
    imputer = SimpleImputer(strategy='mean')  # Hoặc sử dụng 'median' hay 'most_frequent'
    X = imputer.fit_transform(X)  # Điền giá trị NaN

    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    sse.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X, kmeans.labels_))

plt.figure(figsize=(12, 5))
```

- Phương pháp Elbow để xác định SSE

```
# Phương pháp Elbow để xác định SSE
plt.subplot(1, 2, 1)
plt.plot(range(2, 10), sse, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title("Elbow Method")
```

- Đánh giá Silhouette Score

```
# Đánh giá Silhouette Score
plt.subplot(1, 2, 2)
plt.plot(range(2, 10), silhouette_scores, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Score")
plt.show()
```

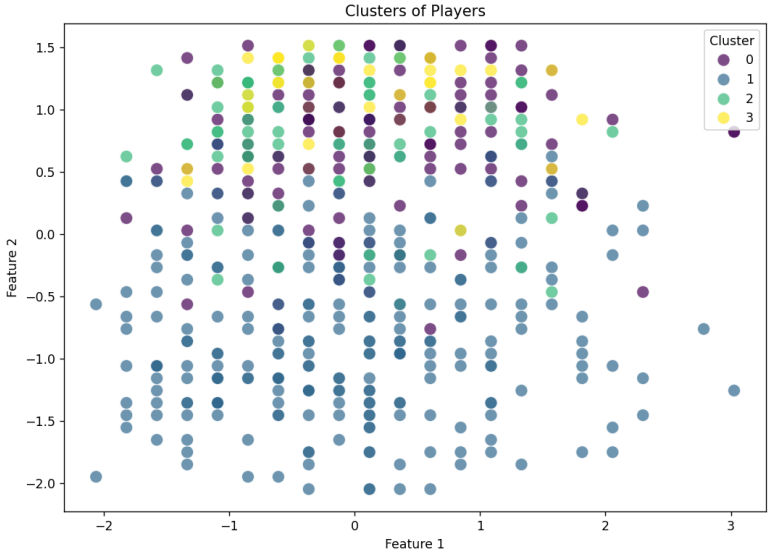


⇨ *Sử dụng số cụm = 4 dựa trên kết quả từ đồ thị*

```
# Sử dụng số cụm = 4 dựa trên kết quả từ đồ thị
optimal_k = 4
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['Cluster'] = kmeans.fit_predict(X)
```

- Trực quan hóa các cụm

```
# Trực quan hóa các cụm
plt.figure(figsize=(10, 7))
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=df['Cluster'], palette='viridis', s=100, alpha=0.7)
plt.title('Clusters of Players')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend(title='Cluster')
plt.show()
```



⇨ **Nhận xét:**

Dựa vào hai biểu đồ, bạn chọn **số cụm tối ưu là 4**, có thể vì:

- Điểm "khuỷu tay" trong biểu đồ SSE xuất hiện xung quanh **k=4**, sau đó SSE giảm chậm lại.
- Silhouette Score tại k=4 có thể đạt mức cao tương đối, cho thấy các cụm khá rõ ràng.
- Cân đối về mặt dữ liệu, giúp phân nhóm mà vẫn giữ sự khác biệt rõ ràng giữa các cụm.

2. Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D

- Giảm số chiều dữ liệu xuống 2 chiều

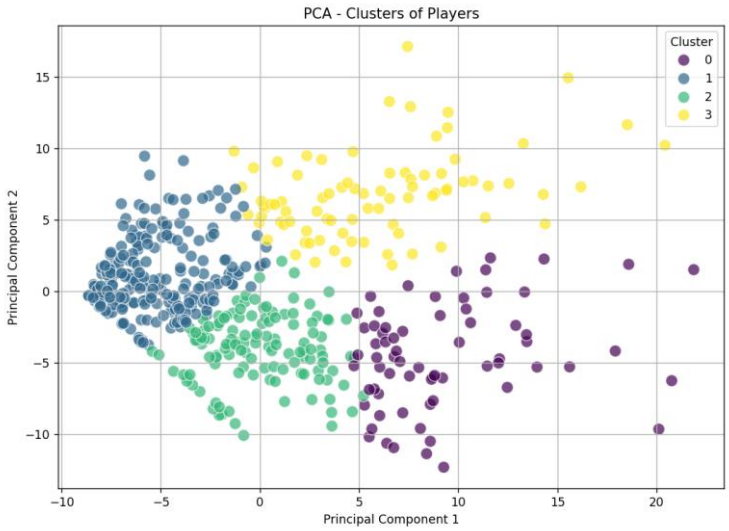
```
# Giảm số chiều dữ liệu xuống 2 chiều
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

- Tạo DataFrame với các thành phần chính và cụm

```
# Tạo DataFrame với các thành phần chính và cụm
df_pca = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
df_pca['Cluster'] = df['Cluster']
```

- Vẽ biểu đồ phân cụm

```
# Vẽ biểu đồ phân cụm
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PC1', y='PC2', hue='Cluster', data=df_pca, palette='viridis', s=100, alpha=0.7)
plt.title('PCA - Clusters of Players')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```



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

- Viết hàm create_radar_chart tạo một biểu đồ radar cho một cầu thủ dựa trên các thuộc tính của họ.

```
def create_radar_chart(ax, player, attributes):

    df = pd.read_csv(args.file)

    # Lấy dữ liệu cho cầu thủ
    values = df.loc[df['name'] == player, attributes].values.flatten()

    # Số lượng thuộc tính
    num_vars = len(attributes)

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

    values = np.concatenate((values, [values[0]]))
    angles += angles[:1]

    colors = ['b', 'r', 'g', 'm', 'y']

    ax.set_facecolor('#f0f0f0')

    # Vẽ biểu đồ radar với màu sắc cho từng thuộc tính
    for i in range(num_vars):
        ax.plot(angles[i:i + 2], values[i:i + 2], color=colors[i % len(colors)], linewidth=2)
        ax.fill(angles, values, facecolor=colors[i % len(colors)], alpha=0.25)

    ax.set_yticklabels([])
    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(attributes)

    ax.set_title(f'Radar Chart for: {player}', weight='bold', size='medium', position=(0.5, 1.1),
                 horizontalalignment='center', verticalalignment='center')
```

- Chuẩn bị dữ liệu và danh sách chỉ số cần so sánh

```
parser = argparse.ArgumentParser(description="Draw radar charts to compare players.")
parser.add_argument('--file', type=str, default='D:/Python/BTL/bai3/file/result.csv')
args = parser.parse_args()

# Đọc dữ liệu từ file
df = pd.read_csv(args.file)

# Lấy tên của hai cầu thủ từ hai dòng đầu tiên
player1 = df['name'].iloc[0]
player2 = df['name'].iloc[1]

# Đặt thuộc tính mặc định để so sánh
default_attributes = ['age', 'matches_played', 'PrgP', 'Pass_Cmp', 'Medium_Cmp', 'Pass_Live']
```

- Tạo biểu đồ vào lưu biểu đồ vào file radar_charts.png

```
# Tạo hình ảnh với hai biểu đồ radar
fig, axs = plt.subplots(1, 2, figsize=(16, 8), subplot_kw=dict(polar=True))

# Vẽ biểu đồ cho mỗi cầu thủ
create_radar_chart(axs[0], player1, default_attributes)
create_radar_chart(axs[1], player2, default_attributes)

# Lưu hình ảnh vào tệp PNG
plt.savefig('D:/Python/BTL/bai3/file/radar_charts.png', bbox_inches='tight')
plt.show()
```

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

1. Thư viện

- requests: Thư viện dùng để gửi yêu cầu truy cập đến trang web và lấy dữ liệu trả về.
- BeautifulSoup: Thư viện dùng để xử lý dữ liệu HTML được lấy từ trang web.
- sys: Thư viện cung cấp các hàm điều khiển môi trường thực thi của Python, ở đây được dùng để thiết lập mã hóa UTF-8 cho đầu ra.
- pandas: Thư viện dùng để tạo và thao tác với DataFrame (một cấu trúc dữ liệu dạng bảng).
- time: Thư viện cung cấp các hàm xử lý thời gian, ở đây dùng để tạm dừng chương trình trong 3 giây.
- csv: Module dùng để xử lý file định dạng CSV.

2. Lấy dữ liệu danh sách đội bóng

- Code truy cập vào trang web: https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league/2023-2024.
- Sử dụng BeautifulSoup để phân tích cú pháp HTML của trang web.

- Tìm bảng dữ liệu chứa thông tin các đội bóng (table) dựa vào class của bảng (table table-striped table-hover leaguetable mvp-table ranking-table mb-0).
- Lấy danh sách các thẻ a (thẻ liên kết) bên trong bảng. Mỗi thẻ a đại diện cho một đội bóng, chứa tên đội và đường dẫn chi tiết của đội.
- Lặp qua từng thẻ a, trích xuất tên đội và đường dẫn, lưu trữ vào danh sách teams_data.

3. Lấy dữ liệu cầu thủ

- Lặp qua danh sách các đội bóng (teams_data).
- Đối với mỗi đội, trích xuất tên đội (team_name) và đường dẫn chi tiết trang đội (team_url).
- Truy cập vào trang chi tiết của đội (team_url).
- Tìm bảng dữ liệu chứa thông tin các cầu thủ (table_tmp) dựa vào class của bảng (table table-striped-rowspan ft-table mb-0).
- Lấy danh sách các hàng dữ liệu trong bảng (players).
- Lọc các hàng dữ liệu của cầu thủ (loại bỏ các hàng tiêu đề).
- Đối với mỗi hàng dữ liệu cầu thủ:
    - Lấy tên cầu thủ (player_name) từ thẻ th con của thẻ tr.
    - Lấy giá trị chuyển nhượng ước tính của cầu thủ (player_cost) từ ô cuối cùng của hàng (td cuối cùng).
    - Lưu trữ thông tin tên cầu thủ, tên đội và giá trị chuyển nhượng vào danh sách players_data.
- In ra thông báo hoàn thành việc lấy dữ liệu cầu thủ của mỗi đội.

4. Lưu trữ dữ liệu

- Tạo một DataFrame từ danh sách players_data bằng thư viện pandas. DataFrame có các cột: 'Player' (tên cầu thủ), 'Team' (tên đội), 'Cost' (giá trị chuyển nhượng).
- Lưu trữ DataFrame vào file CSV tên "results4.csv" với mã hóa UTF-8-sig để đảm bảo hiển thị chính xác tiếng Việt.
- In ra thông báo hoàn thành việc lưu trữ dữ liệu.

5. Tạm dừng

- Sử dụng time.sleep(3) để tạm dừng chương trình trong 3 giây giữa các lần truy cập trang web. Điều này nhằm tránh bị chặn truy cập do nghi ngờ hoạt động bot.