HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

BÀI TẬP LỚN PYTHON

Giảng viên: **Kim Ngọc Bách**

Sinh viên: **Nguyễn Văn Đạt**

Mã sinh viên: **B22DCCN199**

Lớp: **D22CQCN07-B**

Hà Nội, 2024

MỤC LỤC

Bài 1: Viết chương trình Python thu thập dữ liệu phân tích cầu thủ 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.

1.1 Lấy dữ liệu từ url

- Dữ liệu được lấy từ website có địa chỉ url = '**https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats**'.
- Khi có được url ta phân tích và lấy dữ liệu bằng cách sử sụng các thư viện có sẵn

```
url = 'https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
tables_link = soup.find_all('table')  # Tìm tất cả các bảng
table_link = tables_link[0]
rows = table_link.find_all('tr')
team_links = []
team_names = []
for row in rows:
```

- Requests và BeautifulSoup trong Python để lấy và phân tích cú pháp nội dung HTML từ một URL.
- Sử dụng BeautifulSoup để phân tích cú pháp và tìm các bảng (table) trong trang web với **tables_link[0]** là danh sách chứa các bảng chi tiết về các đội.
- Bảng đầu tiên trong danh sách **tables_link** chứa dữ liệu về các đội bóng, vì vậy ta chọn **table_link = tables_link[0]** để lấy dữ liệu chi tiết.
- **table_link.find_all('tr'):** find_all sẽ tìm tất cả các thẻ <tr> trong table_link và trả về một danh sách các thẻ này. Mỗi phần tử trong danh sách đại diện cho một hàng của bảng.
- **rows = table_link.find_all('tr'):** sẽ là một danh sách các hàng (tức là các thẻ <tr>) trong bảng.
- Sau khi đã có được rows thì bắt đầu tạo ra các danh sách để lưu các liên kết đến trang thông tin chi tiết của từng đội(**team_links**) và tên của các đội (**team_names**).
- Sau đó bắt đầu duyệt qua từng hàng trong rows .

1.2 Tìm liên kết đội bóng và tên đội bóng

- Với mỗi lần duyệt qua từng hàng trong rows ta sẽ lấy ra được tên của đội bóng và liên kết đến trang thông tin chi tiết của từng đội.
- **team_cell:** Dòng này tìm một ô (th) chứa thông tin về đội với class là 'left' và data-stat là 'team'.

- Nếu ô này chứa thẻ <a> với thuộc tính **'href'**, liên kết sẽ được nối với **'https://fbref.com/'** để tạo URL đầy đủ, rồi thêm vào danh sách **team_links**.

```
team_cell = row.find('th', {'class': 'left', 'data-stat': 'team'})
    if team_cell:
        a_tag = team_cell.find('a')
        if a_tag and 'href' in a_tag.attrs:
            link = 'https://fbref.com/' + a_tag['href']
            team_links.append(link)
```

- **team_tag:** Dòng này tìm một ô (th) chứa thông tin về tên đội với thuộc tính data-stat là 'team'.
- Nếu team_tag tồn tại và chứa thẻ <a> thì thêm vào danh sách **team_names**.

```
team_tag = row.find('th', {'data-stat': 'team'})
    if team_tag and team_tag.find('a'):
      team_name = team_tag.find('a').text.strip()
       team_names.append(team_name) # Thêm tên đội vào danh sách
```

1.3 Lưu Trữ và Trích Xuất Chỉ Số Từ Các Đội Bóng

- Sau khi có thông tin về các đội và địa chỉ liên kết đến trang thì tạo ra các danh danh sách lưu trữ các chỉ số cần thu thập gồm 10 danh sách.

```
players_data = []
goalkeeping_data = []
shooting_data = []
passing_data = []
passtype_data = []
goal_shot_data = []
defensive_data = []
playing_time_data = []
miscellaneous_data = []
possession_data = []
```

- Sử dụng vòng lặp để lặp qua các đường dẫn dẫn đến thông tin chi tiết về từng thành viên trong đội đó.
- Với mỗi lần lặp thì cần Delay thời gian để không bị chặn truy cập và tìm tất cả các table có trong đường dẫn vì các chỉ số cần thu thập nằm trong 10 table.

```
for link in team_links:
    response = requests.get(link)
    time.sleep(4) # Delay thoi gian khong thi bi chan
    soup = BeautifulSoup(response.text, 'html.parser')
    tables = soup.find_all('table')  # Tìm tất cả các bảng
```

- Với **table = tables[0]** ở đây thì nó chứa các chỉ số cần thu thập bao gồm:
    - Nation, Team, Position, Age
    - Playing time: matches played, starts, minutes
    - Performance: non-PenaltyGoals, PenaltyGoals, Assists, YellowCards,RedCards
    - Expected: xG, npxG, xAG,
    - Progression: PrgC, PrgP, PrgR
    - Per 90 minutes: Gls, Ast, G+A, G-PK, G+A-PK, xG, xAG, xG+xAG, npxG, npxG + xAG
- Với mỗi bảng cần kiểm tra số phút thi đấu của cầu thủ.
- Điều kiện này đảm bảo chỉ thu thập thông tin của cầu thủ đã thi đấu tương đương ít nhất 90 phút. Cột minutes_90s chứa số lần cầu thủ chơi trọn vẹn 90 phút chia cho 90, nên cầu thủ cần có giá trị từ 1 trở lên để được xét vào danh sách.

```
table = tables[0]
minutes = row.find('td', {'data-stat': 'minutes_90s'})
if minutes and minutes.text.strip() and float(minutes.text.strip()) >= 1
```

- Sau đó thu thập các thông tin và chỉ số cần thiết. Chỉ số nào không có hoặc không áp dụng đề là ‘N/a’ và khi thu thập xong thì lưu vào danh sách tương ứng.

```
player_info = {
    'Player': row.find('th', {'data-stat': 'player'}).text.strip() ,
    'Nation': row.find('td', {'data-stat': 'nationality'}).text.strip().split()[-1],
    'Team': team_names[i],
    'Pos': row.find('td', {'data-stat': 'position'}).text.strip(),
    'Age': row.find('td', {'data-stat': 'age'}).text.strip(),

    # Playing time
    'matches_played': row.find('td', {'data-stat': 'games'}).text.strip() or 'N/a',
    'starts': row.find('td', {'data-stat': 'games_starts'}).text.strip() or 'N/a',
    'minutes': row.find('td', {'data-stat': 'minutes'}).text.strip() or 'N/a',
```

- Sau đó thay thế table bằng lần lượt các table có các chỉ số cần thu thập cần thiết nằm trong tables:

```
table = tables[2]  # Bảng Goalkeeping
minutes = row.find('td', {'data-stat': 'minutes_90s'})
if minutes and minutes.text.strip() and float(minutes.text.strip()) >= 1
```

- Với **table = tables[2]** chứa các chỉ số về **Goalkeeping** bao gồm:
    - Performance: GA, GA90, SoTA, Saves, Save%, W, D, L, CS, CS%
    - Penalty Kicks: PKatt, PKA, PKsv, PKm, Save%
- Sau đó thu thập chỉ số tương tự như **table = tables[0]** và lần lượt làm với các table = tables[4], tables[5], tables[6], tables[7], tables[8], tables[9], tables[10] và tables[11].

1.4 Tổng hợp dữ liệu

- Đầu tiên, chúng ta sẽ tạo các DataFrame từ các danh sách dữ liệu đã thu thập trước đó.
- Mỗi DataFrame sẽ chứa thông tin về một khía cạnh khác nhau của cầu thủ:

```
player = pd.DataFrame(players_data)
goal = pd.DataFrame(goalkeeping_data)
shoot = pd.DataFrame(shooting_data)
passing = pd.DataFrame(passing_data)
passtype = pd.DataFrame(passtype_data)
goal_shot = pd.DataFrame(goal_shot_data)
defensive = pd.DataFrame(defensive_data)
playing_time = pd.DataFrame(playing_time_data)
miscellaneous = pd.DataFrame(miscellaneous_data)
possession = pd.DataFrame(possession_data)
```

- Sau đó tạo một danh sách chứa tất cả các DataFrame cần gộp và sau đó thực hiện gộp dữ liệu dựa trên các cột chung như 'Player', 'Nation', 'Team', 'Pos', và 'Age'.

```
dataframes = [goal, shoot, passing, passtype, goal_shot, defensive,
possession, playing_time, miscellaneous]
merged_df = player
for df in dataframes:
    merged_df = pd.merge(merged_df, df, on=["Player", "Nation", "Team",
"Pos", "Age"], how="left")
```

- Sau khi gộp xong, chúng ta sẽ tách phần tên đầu tiên từ cột 'Player' để dễ dàng sắp xếp sau này:

```
merged_df['First Name'] = merged_df['Player'].str.split().str[0]
sorted_df = merged_df.sort_values(by=['First Name', 'Age'],
ascending=[True, False])
```

- Sau đó sắp xếp DataFrame theo 'First Name' và 'Age', với tên được sắp xếp theo thứ tự tăng dần và tuổi theo thứ tự giảm dần:
- Cuối cùng, chúng ta sẽ xóa cột 'First Name' vì nó không còn cần thiết trong kết quả cuối cùng và điền các giá trị rỗng là ‘N/a’.

```
sorted_df = sorted_df.drop(columns=['First Name'])
sorted_df = sorted_df.fillna("N/a")
sorted_df.to_csv('D:/results.csv', index=False, encoding='utf-8')
```

- Cuối cùng, DataFrame đã sắp xếp sẽ được ghi ra file results:

Bài 2: Phân Tích Chỉ Số Cầu Thủ và Đội Bóng Giải Ngoại Hạng Anh 2023-2024

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

- Sau khi có dữ liệu file từ câu 1 là results.csv thì đọc file CSV và chuyển thành DataFrame để dễ dàng phân tích
- Tạo danh sách colum chứa các cột chỉ số cần phân tích (loại bỏ các cột thông tin không phải chỉ số như 'Player', 'Nation', 'Team', 'Pos', 'Age').
- Duyệt qua từng cột trong colum và chuyển đổi các giá trị về kiểu số. Nếu có lỗi chuyển đổi, giá trị sẽ được gán là NaN để xử lý dễ dàng hơn sau đó.
- Sau đó tạo từ điển top_players để lưu kết quả các cầu thủ có điểm cao nhất và thấp nhất cho từng chỉ số.

```
# Đọc file CSV và chuyển thành DataFrame
df = pd.read_csv('D:/results.csv')
colum = [col for col in df.columns if col not in ['Player', 'Nation',
'Team', 'Pos', 'Age']]
for col in colum:
    df[col] = pd.to_numeric(df[col], errors='coerce')
top_players = {'Top 3 High': {}, 'Top 3 Low': {}}
```

- Duyệt qua từng chỉ số trong colum.
- Nếu một cột có toàn giá trị là NaN, bỏ qua.
- Với mỗi cột chỉ số:
    - Dùng nlargest(3, col) để lấy top 3 cầu thủ có giá trị cao nhất.
    - Dùng nsmallest(3, col) để lấy top 3 cầu thủ có giá trị thấp nhất.
- Lưu các kết quả vào từ điển top_players.
- df[['Player', col]] giữ lại cả hai cột Player (tên cầu thủ) và col (chỉ số hiện tại).

```
for col in colum:
    # Bỏ qua cột nào có toàn giá trị NaN sau khi chuyển đổi
    if df[col].notna().sum() == 0:
        continue

    # Lấy top 3 cao nhất cho chỉ số
    top_high = df[['Player', col]].nlargest(3, col)

    # Lấy top 3 thấp nhất cho chỉ số
    top_low = df[['Player', col]].nsmallest(3, col)

    top_players['Top 3 High'][col] = top_high
    top_players['Top 3 Low'][col] = top_low
```

- Kết quả như sau:

```
            Player  matches_played
33     André Onana              38
63      Bernd Leno              38
86  Carlton Morris              38


            Player  matches_played
290  Loris Karius               1
13     Alex Iwobi               2
190    Ionuț Radu               2
```

Top 3 cầu thủ có chỉ số cao nhất và thấp nhất ở matches_played

2.2 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.

- Tạo danh sách colum chứa tên các cột mà không bao gồm 'Player', 'Nation', 'Team', 'Pos', và 'Age'.
- Duyệt qua từng cột trong colum, chuyển đổi dữ liệu của cột thành kiểu số bằng cách:
  - Chuyển đổi thành kiểu chuỗi (astype(str)).
  - Loại bỏ dấu phẩy (str.replace(',', '')).
  - Sử dụng pd.to_numeric() để chuyển đổi thành số, với errors='coerce' để biến các giá trị không thể chuyển đổi thành NaN.
- Tạo một DataFrame all_stats chứa các thống kê (median, mean, std) cho tất cả các cột trong colum.
- Sử dụng dictionary comprehension để tính toán các giá trị thống kê cho từng cột.
- Thêm cột 'Team' với giá trị 'all' vào đầu DataFrame để chỉ định rằng các thống kê này là cho toàn giải đấu.

```
df = pd.read_csv('D:/results.csv')
colum = [col for col in df.columns if col not in ['Player', 'Nation',
'Team', 'Pos', 'Age']]
for col in colum:
    df[col] = pd.to_numeric(df[col].astype(str).str.replace(',', ''),
errors='coerce')
all_stats = pd.DataFrame(
    {f'Median of {col}': [df[col].median()] for col in colum}|
    {f'Mean of {col}': [df[col].mean()] for col in colum}|
    {f'Std of {col}': [df[col].std()] for col in colum}
    )
all_stats.insert(0, 'Team', 'all')
```

- Sử dụng groupby() để nhóm dữ liệu theo cột 'Team', sau đó sử dụng agg() để tính toán median, mean và std cho từng cột trong colum. reset_index() để đưa chỉ số về dạng chuẩn.
- Kết hợp all_stats với team_stats: Sử dụng pd.concat() để ghép hai DataFrame all_stats và team_stats thành một DataFrame duy nhất merged_df. Tham số ignore_index=True giúp đánh số lại các chỉ số trong DataFrame kết quả.
- Sử dụng to_csv() để lưu DataFrame merged_df vào file CSV

```
team_stats = df.groupby('Team')[colum].agg(['median', 'mean',
'std']).reset_index()

team_stats.columns = ['Team'] + [f'{stat.capitalize()} of {col}' for col
in colum for stat in ['median', 'std', 'mean']]

merged_df = pd.concat([all_stats, team_stats], ignore_index=True)
merged_df.to_csv('D:/results2.csv')
```

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

- Sử dụng pd.read_csv() để đọc dữ liệu từ file CSV và lưu vào một DataFrame có tên là df.
- Tạo danh sách colum chứa tên các cột mà không bao gồm 'Player', 'Nation', 'Team', 'Pos', và 'Age'.
- Vẽ histogram cho toàn giải đấu:
    - Vòng lặp: Duyệt qua từng cột trong colum để vẽ histogram cho mỗi chỉ số.
    - plt.figure(figsize=(10, 6)): Tạo một hình mới với kích thước 10x6 inch.
    - sns.histplot(df[col].dropna(), kde=True): Vẽ histogram cho cột hiện tại. dropna() được sử dụng để loại bỏ các giá trị thiếu (NaN) trước khi vẽ.
    - Tham số kde=True thêm đường phân phối xác suất (kernel density estimate) vào biểu đồ.
    - plt.title(), plt.xlabel(), plt.ylabel(): Đặt tiêu đề cho biểu đồ, nhãn cho trục x (chỉ số) và nhãn cho trục y (tần suất).
    - plt.show(): Hiển thị biểu đồ.

```
df = pd.read_csv('D:/results.csv')
colum = [col for col in df.columns if col not in ['Player', 'Nation',
'Team', 'Pos', 'Age']]
for col in colum:
    plt.figure(figsize=(10, 6))
    sns.histplot(df[col].dropna(), kde=True)
    plt.title(f'Distribution of {col} for All Players')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()
```

- Vẽ histogram cho từng đội:

- for team, group in df.groupby('Team'): Nhóm dữ liệu theo cột 'Team'. team là tên đội, và group là DataFrame chứa dữ liệu của đội đó.
- Vòng lặp: Duyệt qua từng cột trong colum để vẽ histogram cho mỗi chỉ số của từng đội.
- Quy trình vẽ biểu đồ tương tự như phần trước, chỉ khác ở tiêu đề và dữ liệu được vẽ (dữ liệu của đội hiện tại group).

```
for team, group in df.groupby('Team'):
    for col in colum:
        plt.figure(figsize=(10, 6))
        sns.histplot(group[col].dropna(), kde=True)
        plt.title(f'Distribution of {col} for Team {team}')
        plt.xlabel(col)
        plt.ylabel('Frequency')
        plt.show()
```

2.4 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

- Tạo danh sách colum chứa tên các cột mà không bao gồm 'Player', 'Nation', 'Team', 'Pos', và 'Age'.
- Duyệt qua từng cột trong colum, và chuyển đổi dữ liệu của cột thành kiểu số
- Sử dụng groupby('Team') để nhóm dữ liệu theo cột 'Team' và tính trung bình (mean()) cho các chỉ số trong colum. Kết quả là một DataFrame mới team_means, trong đó mỗi hàng tương ứng với một đội và các cột chứa giá trị trung bình của các chỉ số.
- Sử dụng idxmax() để tìm tên đội có điểm số cao nhất cho mỗi chỉ số. Kết quả là một Series top_teams, trong đó chỉ số là tên các chỉ số và giá trị là tên đội bóng có điểm số cao nhất cho chỉ số đó.

```
df = pd.read_csv('D:/results.csv')
colum = [col for col in df.columns if col not in ['Player', 'Nation',
'Team', 'Pos', 'Age']]
for col in colum:
    df[col] = pd.to_numeric(df[col].astype(str).str.replace(',', ''),
errors='coerce')
team_means = df.groupby('Team')[colum].mean()
top_teams = team_means.idxmax()
```

- Để xem đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024 thì em sẽ kiểm tra xem đội nào có số các chỉ số cao nhất là nhiều nhất
- team_means.max() tìm giá trị tối đa trong từng cột của DataFrame team_means, tức là tìm điểm số cao nhất cho mỗi chỉ số.
- team_means == team_means.max() tạo ra một DataFrame boolean với giá trị True cho những ô mà giá trị tương ứng bằng điểm số cao nhất và False cho những ô còn lại.

- .sum(axis=1) tính tổng số lượng True (hoặc điểm số cao nhất) cho từng hàng (từng đội). Kết quả sẽ là một Series, trong đó chỉ ra số lượng chỉ số mà mỗi đội có được giá trị cao nhất.
- highest_counts.idxmax() tìm chỉ số của đội có số lượng chỉ số cao nhất (tức là hàng có giá trị lớn nhất trong Series highest_counts). Chỉ số này tương ứng với tên đội có nhiều chỉ số cao nhất. highest_counts.max() tìm giá trị tối đa trong Series highest_counts, tức là số lượng chỉ số mà đội có điểm số cao nhất.

```
highest_counts = (team_means == team_means.max()).sum(axis=1)
max_highest_count_team = highest_counts.idxmax()
max_count = highest_counts.max()
```

- **Kết quả :** Đội có nhiều chỉ số cao nhất là: Manchester City với 63 chỉ số.

➔ Vậy đội Manchester City là đội có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024.

Bài 3: Phân Tích Chỉ Số Cầu Thủ bằng phân cụm K-means và Biểu đồ Radar

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

```
data = pd.read_csv('D:/results.csv')
numeric_data = data.select_dtypes(include=[np.number]).dropna(axis=1)
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numeric_data)
```

- **data = pd.read_csv('D:/results.csv')**: Đọc dữ liệu từ file results.csv trên ổ đĩa D.
- **numeric_data = data.select_dtypes(include=[np.number]).dropna(axis=1)**: Lọc ra các cột chứa số để phục vụ phân cụm. Các cột không phải là số hoặc chứa giá trị thiếu sẽ bị loại bỏ.
- Chuẩn hóa (**Standardization**) là bước quan trọng để đảm bảo rằng tất cả các chỉ số đều nằm trong cùng một thang đo. Các chỉ số sẽ được chuẩn hóa sao cho có trung bình bằng 0 và độ lệch chuẩn bằng 1.
- **scaled_data**: Dữ liệu đã được chuẩn hóa, sẵn sàng để đưa vào thuật toán K-means.

```
distortions = []
K = range(1, 11)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    distortions.append(sum(np.min(cdist(scaled_data,
kmeans.cluster_centers_, 'euclidean'), axis=1)) / scaled_data.shape[0])
plt.figure(figsize=(8, 5))
plt.plot(K, distortions, 'bo-')
plt.xlabel('Số cụm K')
plt.ylabel('Distortion')
plt.title('Elbow Method để tìm số cụm tối ưu')
plt.show()
```

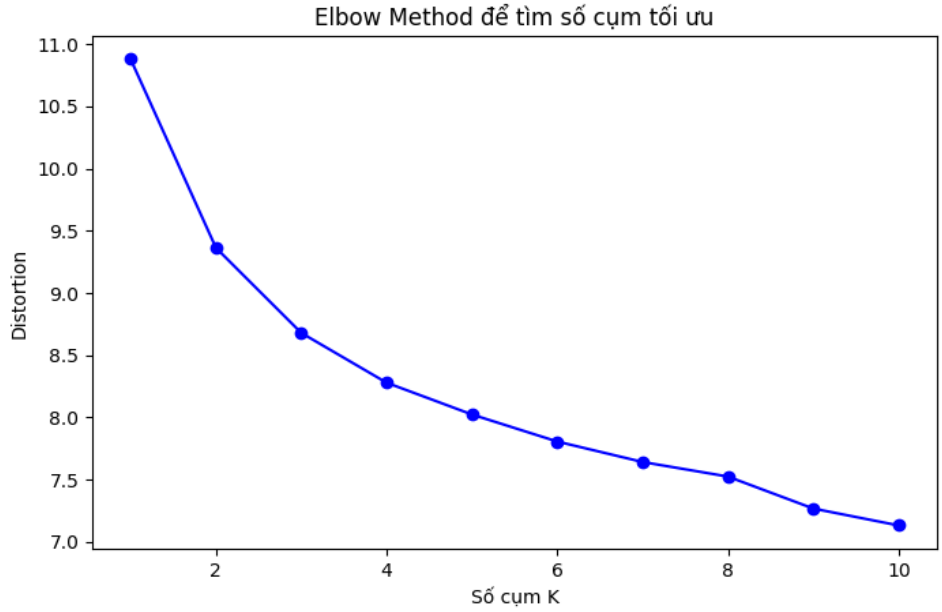
- **Elbow Method** là một phương pháp dùng để tìm số cụm hợp lý bằng cách thử nghiệm với các số cụm khác nhau (ở đây từ 1 đến 10).
- **for k in K**: Chạy thuật toán K-means với số cụm từ 1 đến 10.
- **distortions.append(...)**: Tính toán Distortion (tổng khoảng cách của các điểm trong một cụm đến tâm cụm) cho mỗi giá trị của k, sau đó lưu vào danh sách distortions.
- Sau đó ta vẽ biểu đồ **Elbow** để giúp xác định số cụm tối ưu.

```
optimal_k = 3  # Thay giá trị này sau khi quan sát biểu đồ Elbow
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
data['Cluster'] = kmeans.fit_predict(scaled_data)
```

- **optimal_k**: Chọn số cụm tối ưu sau khi quan sát biểu đồ Elbow (ở đây tạm chọn 3 cụm).
- **data['Cluster'] = kmeans.fit_predict(scaled_data)**: Thực hiện phân cụm với số cụm là optimal_k và gán nhãn cụm cho từng cầu thủ vào cột Cluster.

3.2 Theo bạn thì nên phân loại cầu thủ thành bao nhiêu nhóm? Vì sao? Bạn có Nhận xét gì về kết quả

- **Theo biểu đồ Elbow:**



- Trong biểu đồ, ta thấy một "điểm gấp" (elbow) với k = 3 nơi mà giá trị distortion giảm chậm lại, thì đó có thể là số cụm hợp lý.

➔ Vì vậy k = 3 là lựa chọn hợp lí.

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

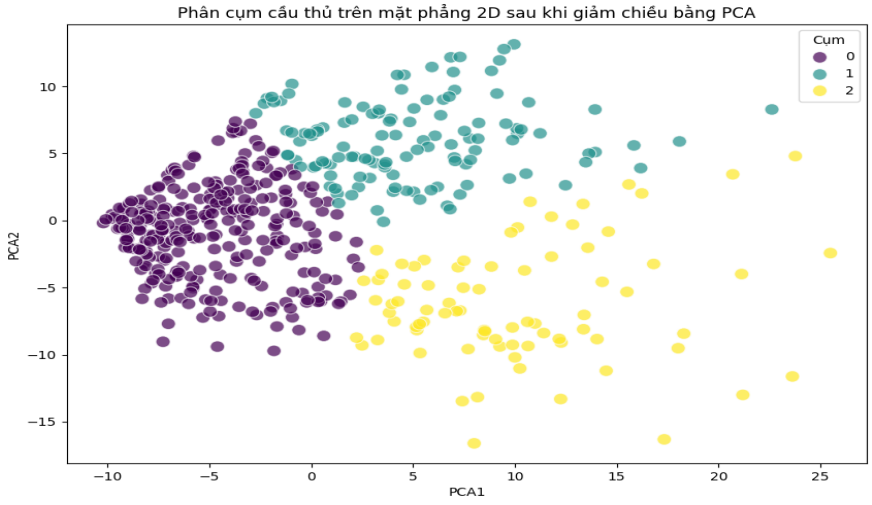
```
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)
data['PCA1'] = pca_data[:, 0]
data['PCA2'] = pca_data[:, 1]
```

- **PCA** (Principal Component Analysis) là một kỹ thuật thống kê được sử dụng để giảm chiều dữ liệu trong khi vẫn giữ lại càng nhiều thông tin càng tốt.
- Trong đoạn mã này, PCA(n_components=2) chỉ định rằng chúng ta muốn giảm chiều dữ liệu xuống còn 2 thành phần chính.
- **fit_transform**(scaled_data) thực hiện việc điều chỉnh PCA đến dữ liệu đã được chuẩn hóa (scaled_data) và trả về một mảng mới có kích thước tương ứng với 2 chiều.
- Sau khi thực hiện **PCA**, chúng ta thêm hai cột mới vào DataFrame gốc data, tương ứng với 2 thành phần chính:
    - PCA1: Thành phần chính đầu tiên.
    - PCA2: Thành phần chính thứ hai.

```
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', data=data,
palette='viridis', s=100, alpha=0.7)
plt.title('Phân cụm cầu thủ trên mặt phẳng 2D sau khi giảm chiều bằng PCA')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.legend(title='Cụm')
plt.show()
```

- **plt.figure(figsize=(10, 7)):** Tạo một hình mới với kích thước 10x7 inch.
- **sns.scatterplot(...)**: Sử dụng Seaborn để vẽ biểu đồ phân tán (scatter plot):
    - **x='PCA1' và y='PCA2'**: Xác định trục hoành và trục tung lần lượt là các thành phần chính vừa tạo ra.
    - **hue='Cluster'**: Chỉ định rằng các điểm trong biểu đồ sẽ được tô màu theo các cụm mà cầu thủ thuộc về.
    - **palette='viridis'**: Chọn bảng màu 'viridis' làm biểu đồ trực quan, dễ phân biệt.
    - **s=100**: Kích thước của các điểm trong biểu đồ.
    - **alpha=0.7**: Độ trong suốt của các điểm, cho phép dễ dàng nhìn thấy các điểm chồng lên nhau.
    - **plt.legend(title='Cụm'):** Thêm chú giải với tiêu đề là 'Cụm' để giúp người xem hiểu được màu sắc tương ứng với từng cụm.
    - **plt.show()**: Hiển thị biểu đồ.

- **Kết quả:**



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

```
def create_radar_chart(player1_data, player2_data, attributes):
    num_vars = len(attributes)
    angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
    player1_data = np.concatenate((player1_data, [player1_data[0]]))
    player2_data = np.concatenate((player2_data, [player2_data[0]]))
    angles += angles[:1]
    fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
    ax.fill(angles, player1_data,color='blue',alpha=0.25,label='Cầu thủ 1')
    ax.fill(angles, player2_data,color='orange',alpha=0.25,label='Cầu thủ
2')
    ax.set_yticklabels([])
    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(attributes)
    plt.title('So sánh cầu thủ')
    plt.legend(loc='upper right')
    plt.show()
```

- Hàm này dùng để vẽ biểu đồ radar cho hai cầu thủ. player1_data và player2_data là các mảng chứa dữ liệu của từng cầu thủ cho các thuộc tính được chọn. attributes là danh sách tên các thuộc tính sẽ được hiển thị trên biểu đồ radar.
- **num**_vars là số lượng thuộc tính được so sánh (số trục trên biểu đồ radar).
- **angles** là danh sách các góc tương ứng với từng thuộc tính, được tính toán để phân bổ đều xung quanh vòng tròn.
- Để vẽ biểu đồ radar dưới dạng một vòng tròn khép kín, điểm đầu tiên của dữ liệu được thêm vào cuối mảng player1 data và player2 data.
- **fig, ax** tạo một biểu đồ phân cực (polar=True), phù hợp cho biểu đồ radar.
- **ax.fill()** vẽ vùng màu đại diện cho dữ liệu của mỗi cầu thủ. **color** và **alpha** xác định màu sắc và độ trong suốt của vùng tô.
- **ax.set_yticklabels([])**: Ẩn nhãn trên trục y (độ lớn của các giá trị không hiển thị).
- **ax.set_xticks(angles[:-1])** và **ax.set_xticklabels(attributes)**: Đặt nhãn cho mỗi thuộc tính trên trục.
- **plt.title và plt.legend**: Đặt tiêu đề và chú thích cho biểu đồ radar.

```
data = pd.read_csv('D:/results.csv')
player1_name = data['Player'].iloc[0]
player2_name = data['Player'].iloc[1]
attributes = ['matches_played', 'minutes', 'starts', 'ProDist',
'TotDist_y', 'Live_x', 'Live_y', 'Touches', 'Def_3rd_y', 'Mn/Start',
'Rec', 'Carries', 'Cmp_y']
player1_data = data.loc[data['Player'] == player1_name,
attributes].values.flatten()
player2_data = data.loc[data['Player'] == player2_name,
attributes].values.flatten()
create_radar_chart(np.array(player1_data, dtype=float),
np.array(player2_data, dtype=float), attributes)
```

- Đọc dữ liệu từ tệp results.csv.
- **attributes** là danh sách các thuộc tính sẽ được so sánh giữa hai cầu thủ.
- **player1_data** và **player2_data** lấy dữ liệu từ tệp CSV cho từng cầu thủ tương ứng, dựa trên tên của họ và các thuộc tính đã chọn.
- **values.flatten()** chuyển đổi dữ liệu thành một mảng 1 chiều.
- Gọi hàm create_radar_chart để vẽ biểu đồ so sánh hai cầu thủ dựa trên các thuộc tính đã chọn.
- **Kết quả:**