Câu 1: Thu thập và xử lý dữ liệu cầu thủ tại giải Ngoại hạng Anh mùa 2023-2024

1. Giới thiệu

Mục tiêu của bài toán là xây dựng một chương trình Python để tự động thu thập và xử lý dữ liệu thống kê của các cầu thủ thi đấu hơn 90 phút trong mùa giải Ngoại hạng Anh 2023-2024 từ trang web **fbref.com**. Dữ liệu cần được thu thập, lọc và lưu trữ vào file results.xlsx, bao gồm các thông tin quan trọng như quốc tịch, đội bóng, vị trí, tuổi, thời gian thi đấu, và các chỉ số liên quan đến hiệu suất thi đấu.

Bài toán yêu cầu sắp xếp dữ liệu theo thứ tự cụ thể và đổi tên các cột sao cho khớp với các tiêu đề mà đề bài đưa ra.

2. Công cụ và thư viện sử dụng

Để hoàn thành bài toán, chương trình sử dụng các thư viện Python sau:

- **Selenium**: Để điều khiển trình duyệt tự động truy cập trang dữ liệu.
- **BeautifulSoup**: Để phân tích HTML và trích xuất dữ liệu từ trang web.
- **Pandas**: Để xử lý dữ liệu dưới dạng DataFrame, giúp thao tác, lọc và sắp xếp dữ liệu dễ dàng.
- **Openpyxl**: Để lưu dữ liệu vào file Excel và căn chỉnh bố cục dữ liệu cho rõ ràng, dễ đọc.
- **Pathlib**: Để quản lý các đường dẫn và tệp tin.

3. Quy trình thực hiện

Quy trình bao gồm các bước: khởi tạo trình duyệt và truy cập trang web, trích xuất và xử lý dữ liệu, đổi tên các cột theo yêu cầu và lưu dữ liệu vào file Excel.

Bước 1: Khởi tạo trình duyệt và truy cập trang web

Sử dụng Selenium để mở trình duyệt Chrome, truy cập trang thống kê dữ liệu cầu thủ của giải Ngoại hạng Anh 2023-2024 trên fbref.com, và tải nội dung trang.

```
url = "https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats"

driver = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))

driver.get(url)

driver.implicitly_wait(10)

page_content = driver.page_source

driver.quit()
```

Bước 2: Phân tích và trích xuất dữ liệu từ trang web

Sau khi tải mã HTML của trang, chương trình sử dụng BeautifulSoup để tìm bảng dữ liệu stats_standard chứa thông tin cầu thủ và trích xuất các hàng có dữ liệu thực tế.

```
soup = BeautifulSoup(page_content, 'html.parser')

table = soup.find('table', {'id': 'stats_standard'})


if table:
    players_data = []
    for row in table.find_all('tr', {'data-row': True}):
        player_data = {}
        for cell in row.find_all(['th', 'td']):
            stat = cell.get('data-stat')
            value = cell.text.strip()
            csk_value = cell.get('csk')  # Lấy số phút thi đấu của cầu thủ
            if stat == 'minutes':
                player_data['csk_minutes'] = int(csk_value.replace(',', '')) if csk_value else 0
            player_data[stat] = value
        players_data.append(player_data)
```

Bước 3: Lọc và sắp xếp dữ liệu

Dữ liệu sau khi thu thập được chuyển thành DataFrame. Chúng tôi lọc chỉ các cầu thủ có số phút thi đấu lớn hơn 90 và sắp xếp theo họ tên và độ tuổi.

```
players_df = pd.DataFrame(players_data)

players_df['Age'] = pd.to_numeric(players_df['age'], errors='coerce')

filtered_players_df = players_df[players_df['csk_minutes'] > 90]


filtered_players_df['first_name'] = filtered_players_df['player'].apply(lambda x: x.split()[0])

filtered_players_df['last_name'] = filtered_players_df['player'].apply(lambda x: x.split()[-1])

filtered_players_df = filtered_players_df.sort_values(by=['last_name', 'Age'], ascending=[True, True])
```

Bước 4: Đổi tên cột và tổ chức dữ liệu

Chương trình sử dụng một từ điển để đổi tên các cột cho phù hợp với yêu cầu của đề bài. Việc đổi tên này giúp cột dữ liệu của file đầu ra dễ hiểu và theo đúng định dạng yêu cầu.

```
column_mapping = {
    'ranker': 'Rk',
    'player': 'Player',
    'nationality': 'Nation',
    'team': 'Team',
    'position': 'Position',
    'age': 'Age',
    'games': 'Matches played',
    'games_starts': 'Starts',
    'minutes': 'Minutes',
    'goals_pens': 'Non-Penalty Goals',
    'pens_made': 'Penalty Goals',
    'assists': 'Assists',
    'cards_yellow': 'Yellow Cards',
    'cards_red': 'Red Cards',
    'xg': 'xG',
    'npxg': 'npxG',
    'xg_assist': 'xAG',
    'progressive_carries': 'PrgC',
    'progressive_passes': 'PrgP',
    'progressive_passes_received': 'PrgR',
    'goals_per90': 'Gls/90',
    'assists_per90': 'Ast/90',
    'goals_assists_per90': 'G+A/90',
    'goals_pens_per90': 'G-PK/90',
    'goals_assists_pens_per90': 'G+A-PK/90',
```

```
    'xg_per90': 'xG/90',
    'xg_assist_per90': 'xAG/90',
    'xg_xg_assist_per90': 'xG + xAG/90',
    'npxg_per90': 'npxG/90',
    'npxg_xg_assist_per90': 'npxG + xAG/90'
}


filtered_players_df.rename(columns=column_mapping, inplace=True)
```

Mỗi cột được đổi tên theo các chỉ số cần thiết, ví dụ như nationality đổi thành Nation, team đổi thành Team, và minutes đổi thành Minutes để tuân theo yêu cầu của đề bài.

Bước 5: Lưu dữ liệu vào file Excel và căn chỉnh

Sau khi đổi tên và sắp xếp, dữ liệu được lưu vào file Excel results.xlsx. Mỗi ô được căn giữa để đảm bảo dữ liệu rõ ràng và dễ nhìn.

```
wb = Workbook()
ws = wb.active
for r in dataframe_to_rows(filtered_players_df, index=False, header=True):
    ws.append(r)
for row in ws.iter_rows():
    for cell in row:
        cell.alignment = Alignment(horizontal="center", vertical="center")
wb.save('results.xlsx')
```

4. Kết quả và nhận xét

- **Kết quả**: Chương trình thu thập dữ liệu cầu thủ thành công và lưu vào file results.xlsx. File Excel này bao gồm các thông tin quan trọng của các cầu thủ như tên, quốc tịch, đội bóng, vị trí, tuổi, thời gian thi đấu, và các chỉ số thi đấu khác. Các cột được đổi tên đúng chuẩn theo yêu cầu, giúp dữ liệu rõ ràng và dễ đọc.
- **Nhận xét**: Quá trình sử dụng Selenium để truy cập và BeautifulSoup để trích xuất dữ liệu từ HTML là hiệu quả. Sau đó, Pandas hỗ trợ tốt cho việc lọc và sắp xếp dữ liệu, và Openpyxl cho phép xuất dữ liệu ra file Excel với định dạng và bố cục mong muốn.

Câu 2: Phân tích và thống kê dữ liệu cầu thủ tại giải Ngoại hạng Anh mùa 2023-2024

1. Giới thiệu

Mục tiêu của bài tập này là phân tích dữ liệu cầu thủ dựa trên các chỉ số đã thu thập được từ fbref.com. Các bước phân tích bao gồm:

- Xác định top 3 và bottom 3 cầu thủ ở mỗi chỉ số.
- Tính toán trung vị, trung bình, và độ lệch chuẩn cho các chỉ số toàn giải và theo từng đội.
- Vẽ biểu đồ phân phối (histogram) cho từng chỉ số cho toàn bộ cầu thủ và từng đội.
- Xác định đội có phong độ tốt nhất dựa trên số lượng chỉ số cao nhất.

Ý 1: Tìm top 3 và bottom 3 cầu thủ ở mỗi chỉ số

Quy trình thực hiện

1. **Nạp dữ liệu**: Chúng tôi sử dụng thư viện pandas để nạp dữ liệu từ file CSV results.csv. File này chứa dữ liệu cầu thủ, bao gồm các thông tin định tính (như tên cầu thủ, đội bóng, vị trí) và các chỉ số định lượng (số bàn thắng, số thẻ phạt, v.v.).

```
file_path = 'results.csv'

data = pd.read_csv(file_path, header=2)
```

2. **Xác định cột chỉ số**: Dùng select_dtypes(include='number') để chọn các cột có kiểu dữ liệu số, vì chỉ các cột này là chỉ số cần phân tích. Bỏ qua các cột không phải chỉ số như Team, Nation, và Position.

```
numeric_columns = data.select_dtypes(include='number').columns.tolist()

exclude_columns = ['Unnamed: 0', 'Team', 'Nation', 'Position']

metric_columns = [col for col in numeric_columns if col not in exclude_columns]
```

3. **Tìm top 3 và bottom 3**: Với mỗi chỉ số trong metric_columns, chúng tôi sắp xếp cầu thủ theo thứ tự giảm dần để lấy top 3 và thứ tự tăng dần để lấy bottom 3. Kết quả cho mỗi chỉ số bao gồm 3 cầu thủ có điểm cao nhất và 3 cầu thủ có điểm thấp nhất.

```
for column in metric_columns:

    top_3 = data[['Player', column]].sort_values(by=column, ascending=False).head(3)

    bottom_3 = data[['Player', column]].sort_values(by=column, ascending=True).head(3)

    top_bottom_players[column] = {'top_3': top_3, 'bottom_3': bottom_3}
```

4. **Xuất kết quả**: Dữ liệu được lưu vào file results1.txt theo cấu trúc: tên chỉ số, danh sách top 3 cầu thủ và bottom 3 cầu thủ. Điều này giúp dễ dàng tra cứu cầu thủ nào đạt thành tích cao nhất hoặc thấp nhất ở từng chỉ số.

Kết quả và phân tích

- **Kết quả**: File results1.txt chứa các bảng top 3 và bottom 3 cho từng chỉ số, cung cấp góc nhìn chi tiết về các cầu thủ có thành tích nổi bật hoặc kém nhất trong mỗi chỉ số.
- **Phân tích**: Bảng xếp hạng này giúp dễ dàng so sánh hiệu suất của các cầu thủ trong từng chỉ số cụ thể, từ đó nhận diện những cầu thủ xuất sắc hoặc yếu kém nhất trong giải đấu.

Ý 2: Tính toán trung vị, trung bình và độ lệch chuẩn cho toàn bộ giải và từng đội

Quy trình thực hiện

1. **Tính toán trên toàn bộ giải đấu**:
    - Khởi tạo một dictionary stats để lưu trữ kết quả với nhóm đầu tiên là "All" (tất cả cầu thủ trong giải).
    - Sử dụng median(), mean(), và std() trên toàn bộ dữ liệu cho mỗi chỉ số, lưu kết quả vào stats.

```
stats = {'Team': ['All']}

for column in metric_columns:
    stats[f'Median of {column}'] = [data[column].median()]
    stats[f'Mean of {column}'] = [data[column].mean()]
    stats[f'Std of {column}'] = [data[column].std()]
```

2. **Tính toán theo từng đội**:
    - Nhóm dữ liệu theo Team và tính toán trung vị, trung bình, độ lệch chuẩn cho mỗi chỉ số trong từng đội.
    - Kết quả này giúp so sánh hiệu suất của các đội trong từng chỉ số.

```
for team_name, group in grouped:
    team_stats = {'Team': team_name}
    for column in metric_columns:
        team_stats[f'Median of {column}'] = group[column].median()
        team_stats[f'Mean of {column}'] = group[column].mean()
        team_stats[f'Std of {column}'] = group[column].std()
```

3. **Xuất kết quả**: Dữ liệu được tổ chức thành DataFrame và lưu vào file results2.csv.

Kết quả và phân tích

- **Kết quả**: File results2.csv chứa median, mean, và std của từng chỉ số cho tất cả cầu thủ và từng đội. File này hỗ trợ phân tích xu hướng chung của giải và so sánh sự khác biệt về hiệu suất giữa các đội.
- **Phân tích**: Trung vị cho thấy giá trị ở giữa, còn trung bình và độ lệch chuẩn giúp đánh giá tính ổn định và mức độ biến động của các chỉ số trong đội và toàn giải đấu. Đội nào có độ lệch chuẩn thấp hơn thường có sự đồng đều về hiệu suất.

Ý 3: Vẽ biểu đồ phân phối (histogram) cho mỗi chỉ số

Phần a - Phân tích tất cả cầu thủ

1. **Chuẩn bị dữ liệu**: Chọn các cột chỉ số từ dữ liệu của tất cả cầu thủ.
2. **Vẽ histogram**: Sử dụng thư viện matplotlib để vẽ biểu đồ histogram cho từng chỉ số, cho thấy phân phối điểm số của từng chỉ số trên toàn giải.
3. **Lưu hình ảnh**: Tất cả các biểu đồ histogram được lưu vào file all_histograms.png.

```
for i, column in enumerate(numeric_columns):
    axes[i].hist(df[column].dropna(), bins=10, edgecolor='black')
    axes[i].set_title(f'Histogram phân bố cho {column}')
plt.savefig("all_histograms.png")
```

Phần b - Phân tích từng đội

1. **Lọc và vẽ theo từng đội**: Dữ liệu được chia theo Team và vẽ histogram cho từng chỉ số của từng đội. Mỗi đội có một file hình riêng, ví dụ: Liverpool_histograms.png.
2. **Lưu hình ảnh**: Biểu đồ histogram của từng đội được lưu vào các file riêng, dễ dàng so sánh phân phối của các chỉ số giữa các đội.

```
for team_name, group in df.groupby(group_column):
    fig, axes = plt.subplots(n_rows, n_columns, figsize=(15, n_rows * 4))
    for i, column in enumerate(numeric_columns):
        axes[i].hist(group[column].dropna(), bins=10, edgecolor='black')
    filename = f"{team_name}_histograms.png"
    plt.savefig(filename)
```

Kết quả và phân tích

- **Kết quả**: File all_histograms.png chứa biểu đồ phân phối của tất cả cầu thủ, còn mỗi đội có một file histogram riêng.
- **Phân tích**: Biểu đồ phân phối cho thấy các đặc điểm như độ tập trung và sự phân tán của các chỉ số. Ví dụ, nếu histogram có dạng phân phối đều, chỉ số đó có sự phân bố đồng đều giữa các cầu thủ, còn nếu tập trung nhiều ở một phía, chỉ số đó có sự chênh lệch rõ rệt.

Ý 4: Tìm đội bóng có phong độ tốt nhất ở mỗi chỉ số

1. **Đọc dữ liệu từ file CSV**

```
file_path = 'results2.csv'

data = pd.read_csv(file_path)
```

- Đầu tiên, mã code sử dụng thư viện pandas để đọc dữ liệu từ tệp *results2.csv*. Dữ liệu này chứa các chỉ số thống kê của các cầu thủ trong mùa giải.
- file_path là đường dẫn tới file CSV, và dữ liệu được lưu vào biến data dưới dạng một DataFrame để dễ dàng thao tác.

2. **Xác định các cột chứa chỉ số**

```
numeric_columns = data.select_dtypes(include='number').columns.tolist()

exclude_columns = ['Unnamed: 0', 'Team', 'Nation', 'Position']

metric_columns = [col for col in numeric_columns if col not in exclude_columns]
```

- Đoạn mã trên lọc ra các cột chứa dữ liệu số (các cột này đại diện cho các chỉ số cần phân tích).
- numeric_columns chứa danh sách các cột có kiểu dữ liệu số, còn exclude_columns là danh sách các cột không phải chỉ số (chẳng hạn như ‘Team’, ‘Nation’, ‘Position’).
- Cuối cùng, metric_columns giữ lại các cột số đại diện cho chỉ số, sau khi đã loại bỏ các cột không cần thiết.

3. **Tìm đội có giá trị cao nhất ở mỗi chỉ số**

```
team_top_metrics = {}


for column in metric_columns:
    top_team_row = data[['Team', column]].sort_values(by=column, ascending=False).head(1)
    team_name = top_team_row['Team'].values[0]
    max_value = top_team_row[column].values[0]


    team_top_metrics[column] = {
        'Team': team_name,
        'Max_Value': max_value
    }
```

- Tạo một dictionary team_top_metrics để lưu đội có giá trị cao nhất cho mỗi chỉ số.
- Đoạn mã này lặp qua từng chỉ số trong metric_columns:
    - Sắp xếp dữ liệu theo chỉ số đó theo thứ tự giảm dần (ascending=False).
    - Chọn hàng đầu tiên (cầu thủ hoặc đội có giá trị cao nhất ở chỉ số đó).
    - Lưu tên đội và giá trị lớn nhất của chỉ số vào dictionary team_top_metrics.

4. **Đếm số lần mỗi đội đứng đầu các chỉ số**

```
team_performance = {}
for metric, info in team_top_metrics.items():
    team_name = info['Team']
    team_performance[team_name] = team_performance.get(team_name, 0) + 1
```

- Tạo một dictionary team_performance để đếm số lần mỗi đội có giá trị cao nhất ở các chỉ số.
- Đoạn mã này lặp qua từng mục trong team_top_metrics, tăng số đếm cho đội tương ứng trong team_performance.
- Kết quả sẽ là số lượng các chỉ số mà mỗi đội dẫn đầu.

5. **Xác định đội có phong độ tốt nhất**

```
best_team = max(team_performance, key=team_performance.get)

best_team_metrics_count = team_performance[best_team]
```

- best_team là đội có nhiều chỉ số cao nhất bằng cách tìm đội có giá trị lớn nhất trong team_performance.
- best_team_metrics_count là số lượng chỉ số mà đội này dẫn đầu.

6. **Ghi kết quả ra file**

```
with open('results4.txt', 'w', encoding='utf-8') as f:
    f.write("Đội có chỉ số cao nhất ở mỗi chỉ số:\n")
    for metric, info in team_top_metrics.items():
        f.write(f"Chỉ số: {metric}, Đội: {info['Team']}, Giá trị lớn nhất: {info['Max_Value']}\n")

    f.write(f"\nĐội có phong độ tốt nhất: {best_team} với {best_team_metrics_count} chỉ số cao nhất.\n")
```

- Mã code này mở tệp results4.txt để ghi lại kết quả.
- Ghi từng chỉ số kèm đội có giá trị cao nhất, sau đó ghi tên đội có phong độ tổng thể tốt nhất và số lượng chỉ số họ dẫn đầu.

3.1 Phân cụm cầu thủ bằng K-means

Mục tiêu

Mục tiêu của phần này là sử dụng thuật toán K-means để phân chia các cầu thủ thành các nhóm dựa trên đặc điểm thống kê.

Phân tích mã code K-means

1. **Đọc dữ liệu từ file CSV**

```
file_path = 'results.csv'

data = pd.read_csv(file_path, header=2)  # Đọc dữ liệu từ dòng thứ 3 (nếu cần)
```

   - Đọc dữ liệu từ file *results.csv*, trong đó header=2 sẽ bỏ qua hai dòng đầu tiên nếu cần thiết.

2. **Xử lý dữ liệu số và kiểm tra giá trị thiếu**

```
data_numeric = data.select_dtypes(include=['float64', 'int64'])

print("Kiểu dữ liệu của các cột số:")

print(data_numeric.dtypes)
```

```
print("Số lượng giá trị NaN trong các cột số:")
print(data_numeric.isnull().sum())
```

- Lọc các cột có kiểu dữ liệu số để chỉ giữ lại các chỉ số cầu thủ.
- Kiểm tra sự xuất hiện của các giá trị thiếu (NaN), giúp đảm bảo dữ liệu đầy đủ và chính xác.

```
if data_numeric.isnull().sum().sum() > 0:
    data_numeric.fillna(data_numeric.mean(), inplace=True)
```

- Nếu phát hiện giá trị thiếu, mã sẽ thay thế chúng bằng giá trị trung bình của cột tương ứng.

3. **Chuẩn hóa dữ liệu**

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(data_numeric)
```

- Dữ liệu được chuẩn hóa về cùng một thang đo bằng StandardScaler, giúp cải thiện hiệu quả của thuật toán K-means khi xử lý các chỉ số có đơn vị khác nhau.

4. **Áp dụng K-means để phân cụm**

```
k = 4  # Chọn số cụm
kmeans = KMeans(n_clusters=k, random_state=42)
data['Cluster'] = kmeans.fit_predict(X_scaled)
```

- Thuật toán K-means được áp dụng với k=4 cụm để phân chia các cầu thủ thành 4 nhóm chính.
- random_state=42 giúp đảm bảo kết quả phân cụm ổn định khi chạy lại mã.
- Kết quả phân cụm được lưu vào cột mới 'Cluster' của data, giúp dễ dàng nhận diện cụm của từng cầu thủ.

5. **Đặt tên cho các cụm**

```
cluster_names = {0: 'Tiền đạo', 1: 'Tiền vệ', 2: 'Hậu vệ', 3: 'Thủ môn'}
data['Cluster_Name'] = data['Cluster'].map(cluster_names)
```

- Đặt tên cho các cụm, tương ứng với các vị trí "Tiền đạo", "Tiền vệ", "Hậu vệ", và "Thủ môn", giúp làm rõ đặc điểm chính của mỗi cụm.

6. **Xuất kết quả phân cụm ra file**

```
result_txt = data[['Player', 'Cluster', 'Cluster_Name']].to_string(index=False)

with open('kmeans_results.txt', 'w', encoding='utf-8') as file:
    file.write("Kết quả phân cụm với tên:\n")
    file.write(result_txt)
```

- Kết quả phân cụm (gồm tên cầu thủ và cụm của họ) được ghi vào tệp *kmeans_results.txt*, lưu trữ thông tin cần thiết cho việc phân tích và kiểm tra.

Kết quả

Kết quả của phần này là danh sách các cầu thủ được phân loại vào 4 nhóm, mỗi nhóm đại diện cho một vị trí cầu thủ phổ biến trong bóng đá. Điều này giúp hiểu rõ sự khác biệt giữa các nhóm cầu thủ dựa trên các chỉ số thống kê.

Câu 3.2:

Trong bài toán này, việc chọn **4 nhóm** là hợp lý vì trong bóng đá, các vị trí cầu thủ thường được phân chia thành 4 nhóm chính, bao gồm:

- **Tiền đạo**: Tập trung vào các cầu thủ chơi gần khung thành đối phương, nhiệm vụ chính là ghi bàn.
- **Tiền vệ**: Cầu thủ kết nối giữa phòng thủ và tấn công, đảm bảo cả vai trò phòng thủ lẫn kiến tạo.
- **Hậu vệ**: Tập trung vào nhiệm vụ phòng thủ, bảo vệ khung thành đội nhà và ngăn cản đối thủ ghi bàn.
- **Thủ môn**: Cầu thủ đặc biệt với nhiệm vụ chính là bảo vệ khung thành khỏi các cú sút của đối phương.

Vì vậy, **việc chọn 4 cụm tương ứng với các vai trò phổ biến trong bóng đá là hợp lý** và giúp phân tích dữ liệu cầu thủ theo từng vị trí đặc thù của họ.

5.2 Nhận xét về kết quả

- **Tính hợp lý của các nhóm**: Kết quả phân cụm có ý nghĩa thực tế vì các cầu thủ trong cùng một nhóm có xu hướng chia sẻ các đặc điểm tương đồng về thống kê, phản ánh vai trò và nhiệm vụ của họ trong trận đấu.
- **Đánh giá khả năng của K-means**: Thuật toán K-means hoạt động tốt với dữ liệu này, vì các cụm được hình thành dựa trên khoảng cách giữa các điểm dữ liệu và phù hợp khi các nhóm được xác định bởi các vai trò chuyên biệt của cầu thủ.
- **Giá trị thực tiễn**: Kết quả phân cụm giúp phân tích sâu hơn về phong độ và phong cách thi đấu của các cầu thủ. Các huấn luyện viên và nhà phân tích có thể dùng kết quả này để điều chỉnh chiến thuật hoặc tuyển dụng cầu thủ phù hợp với vị trí cần thiết.

Tóm lại, lựa chọn 4 nhóm là hợp lý và phản ánh rõ nét vai trò đặc thù của cầu thủ trong bóng đá. Kết quả phân cụm cung cấp những thông tin hữu ích cho việc đánh giá và phân tích hiệu suất của cầu thủ theo từng vị trí cụ thể.

3.3 Giảm chiều dữ liệu bằng PCA

Mục tiêu

Phần này nhằm giảm số chiều của dữ liệu xuống 2 chiều để trực quan hóa các cụm cầu thủ trên mặt phẳng 2D, giúp dễ dàng nhận diện sự khác biệt giữa các cụm.

Phân tích mã code PCA

1. **Áp dụng PCA để giảm số chiều**

```
pca = PCA(n_components=2)

X_pca = pca.fit_transform(X_scaled)
```

   - Sử dụng phương pháp Phân tích Thành phần Chính (PCA) để giảm chiều của dữ liệu từ nhiều chỉ số xuống chỉ còn hai thành phần chính (PC1 và PC2).
   - Việc giảm chiều giúp giữ lại thông tin quan trọng nhất và đơn giản hóa dữ liệu, phục vụ cho việc trực quan hóa.

2. **Tạo DataFrame mới để lưu trữ kết quả PCA và nhãn cụm**

```
pca_df = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])

pca_df['Cluster'] = data['Cluster']

pca_df['Cluster_Name'] = pca_df['Cluster'].map(cluster_names)
```

   - Một DataFrame pca_df được tạo để lưu kết quả PCA (hai thành phần chính) và nhãn cụm từ kết quả K-means, giúp dễ dàng biểu diễn dữ liệu.

3. **Trực quan hóa các cụm trên mặt phẳng 2D**

```
plt.figure(figsize=(12, 8))

sns.scatterplot(data=pca_df, x='PC1', y='PC2', hue='Cluster_Name', palette='viridis', s=100, alpha=0.8)

plt.title("Phân cụm K-means trên mặt phẳng 2D sau khi giảm số chiều bằng PCA")

plt.xlabel("Tấn công (PC1)")

plt.ylabel("Phòng thủ (PC2)")

plt.legend(title='Cụm')
```

   - Biểu đồ phân tán giúp trực quan hóa sự phân chia các cụm sau khi giảm chiều.

- Các cụm được biểu diễn bằng các màu sắc khác nhau, giúp phân biệt các nhóm cầu thủ trên mặt phẳng 2D. Các nhãn "Tấn công" và "Phòng thủ" được thêm vào các thành phần chính để biểu thị ý nghĩa chính của từng trục.

4. **Lưu biểu đồ vào file ảnh**

```
plt.savefig('pca_kmeans_clusters.png', format='png', dpi=300)

plt.show()
```

- Biểu đồ được lưu vào file *pca_kmeans_clusters.png*, giúp lưu trữ kết quả cho việc phân tích hoặc trình bày.