HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO BÀI TẬP LỚN
Môn LẬP TRÌNH VỚI PYTHON

Giảng viên: Kim Ngọc Bách

Sinh viên	Lê Cao Ngọc Linh
Mã sinh viên	B22DCCN489
Lớp	D22CQCN09-B
Nhóm học phần	11
Số điện thoại	0337538039
Email	LinhLCN.B22CN489@stu.ptit.edu.vn

Hà Nội - 31/11/2024

MỤC LỤC

Giới thiệu chung

Python là một trong những ngôn ngữ lập trình đa năng và phổ biến nhất hiện nay, nổi bật với cú pháp dễ hiểu và một hệ sinh thái thư viện phong phú. Đặc biệt trong lĩnh vực xử lý dữ liệu và khoa học dữ liệu, Python sở hữu các thư viện mạnh mẽ hỗ trợ đắc lực cho quá trình phân tích và trực quan hóa dữ liệu.

Các thư viện chính được sử dụng trong bài tập này bao gồm:

- NumPy: Cung cấp các hàm và công cụ làm việc với mảng (array) và các phép toán đại số tuyến tính, giúp tối ưu hóa hiệu năng trong các tính toán với dữ liệu lớn.
- Pandas: Thư viện giúp dễ dàng làm việc với dữ liệu dạng bảng (DataFrame). Pandas hỗ trợ các thao tác như lọc, nhóm, chuyển đổi dữ liệu một cách nhanh chóng và hiệu quả.
- Matplotlib và Seaborn: Cung cấp các công cụ trực quan hóa dữ liệu mạnh mẽ. Matplotlib cho phép tạo ra các biểu đồ linh hoạt, trong khi Seaborn giúp tạo ra các biểu đồ với giao diện hiện đại và phù hợp với các phân tích thống kê.
- Scikit-learn: Một thư viện machine learning phổ biến, cung cấp các thuật toán học máy, các công cụ tiền xử lý dữ liệu và đánh giá mô hình, hữu ích khi bài toán yêu cầu xây dựng mô hình dự báo.
- Selenium: cho phép tự động hóa các tác vụ trên trình duyệt web. Bạn có thể sử dụng nó để mở trang web, tương tác với các phần tử như nhấp chuột, nhập văn bản, và lấy dữ liệu từ trang web.
- Selenium cho phép tự động hóa các tác vụ trên trình duyệt web. Bạn có thể sử dụng nó để mở trang web, tương tác với các phần tử như nhấp chuột, nhập văn bản, và lấy dữ liệu từ trang web.
- BeautifulSoup là một thư viện Python dùng để phân tích cú pháp HTML và XML. Nó giúp bạn dễ dàng trích xuất dữ liệu từ các tài liệu HTML phức tạp.

Các thư viện này sẽ giúp tối ưu hóa quá trình xử lý và phân tích dữ liệu, đồng thời hỗ trợ tạo ra các biểu đồ trực quan, dễ hiểu để minh họa kết quả phân tích và đưa ra kết luận.

A. Bài 1

I. Phân tích bài toán

Bài 1 yêu cầu viết chương trình thu thập dữ liệu thống kê của các cầu thủ trong Giải Ngoại Hạng Anh 2023-2024 từ trang fbref.com. Cụ thể:

- Tiêu chí lọc: Chỉ thu thập dữ liệu của các cầu thủ có số phút thi đấu trên 90 phút.
- Dữ liệu cần thu thập: Các chỉ số cụ thể về quốc tịch, đội bóng, vị trí, độ tuổi, thời gian thi đấu, hiệu suất, dự đoán, và nhiều số liệu thống kê chi tiết khác được liệt kê theo từng danh mục (như Sút bóng, Chuyền bóng, Kiến tạo, Phòng thủ, và Thời gian thi đấu).
- Đầu ra: Lưu kết quả thu thập được vào tệp results.csv, với cấu trúc:
    - Mỗi cột tương ứng với một chỉ số.
    - Sắp xếp các cầu thủ theo thứ tự tên. Nếu trùng tên, sắp xếp theo tuổi từ lớn đến nhỏ.
    - Nếu có chỉ số không có giá trị, đánh dấu là N/a.

II. Cài đặt và công cụ

Bài 1 cần cài đặt các thư viện sau:

- selenium: để tự động hóa trình duyệt.
- beautifulsoup4: để phân tích cú pháp HTML và XML.
- pandas: để xử lý và phân tích dữ liệu.
- webdriver-manager: để quản lý trình điều khiển cho Selenium.

III. Code

- Import các thư viện cần thiết . Tạo một hàm validdata(n) để chuyển đổi dữ liệu trong khi thu thập từ trang web

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup as bs
import pandas as pd

# Hàm kiểm tra và chuyển đổi dữ liệu
def validdata(n):
    if n == '' or n is None:
        return "N/a"
    try:
        return float(n)
    except ValueError:
        return "N/a"
```

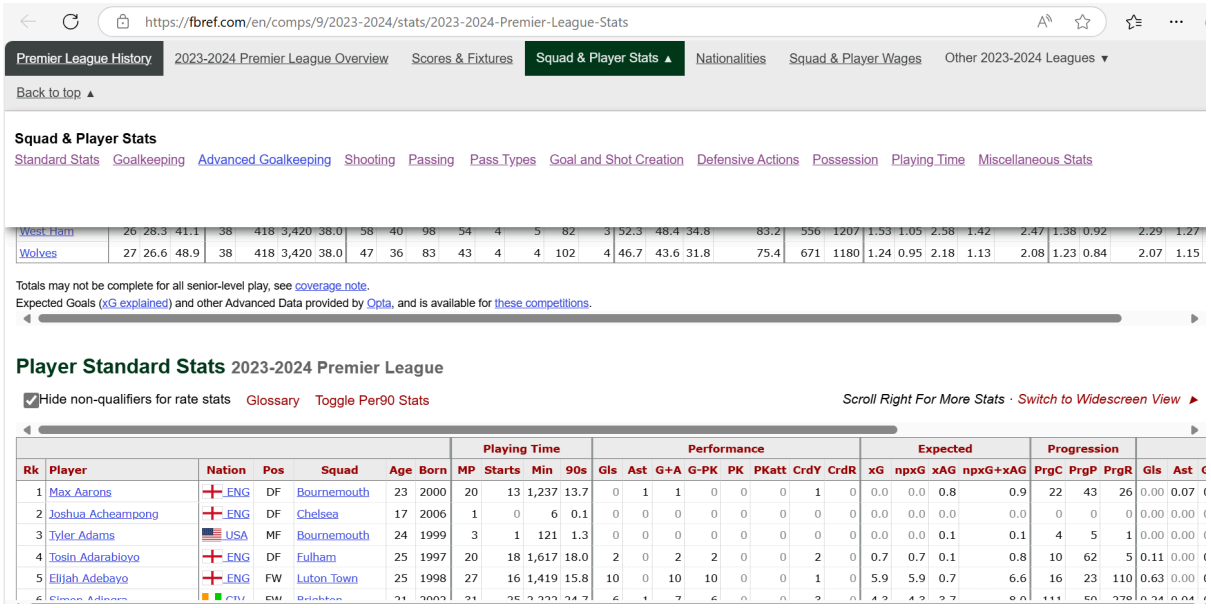
- Tạo hàm thu thập dữ liệu GetDatafromWeb(url, Xpath_player, Data_name): Hàm trả về danh sách người chơi kèm chỉ số đã thu thập được ở từng trang web

```python
# Hàm lấy dữ liệu từ web
def GetDataFromWeb(url, Xpath_player, Data_name):
    # Cài đặt trình điều khiển Chrome
    service = Service(ChromeDriverManager().install())
    driver1 = webdriver.Chrome(service=service)
    # Truy cập trang web
    driver1.get(url)
    headers = []
    sub_headers = []
    player_list = []
    try:
        # Đợi cho đến khi bảng xuất hiện (tối đa 10 giây)
        WebDriverWait(driver1, 10).until(EC.presence_of_element_located((By.XPATH, Xpath_player)))

        # Lấy bảng HTML bằng Selenium với XPath
        table_element = driver1.find_element(By.XPATH, Xpath_player)

        # Lấy mã nguồn HTML của bảng
        html_table = table_element.get_attribute('outerHTML')

        # Phân tích HTML bằng BeautifulSoup
        soup = bs(html_table, 'html.parser')

        # Tìm bảng trực tiếp từ mã HTML đã lấy
        table = soup.find('table')
        if table:
            for row in table.find_all('tr'):
                cols = row.find_all('td')
                data = []
                for id, play in enumerate(cols[:-1]):
                    if id == 1:
                        a = play.text.strip().split()
                        if len(a) == 2:
                            data.append(a[1])
                        else:
                            data.append(play.text.strip())
                    else:
                        s = play.text.strip()
                        if id >= 4:
                            s = s.replace(",", "")
                            s = validdata(s)
                        data.append(s)
                if len(data) != 0: player_list.append(data)
    finally:
        # Đóng trình duyệt sau khi hoàn tất
        driver1.quit()
        print("Finish " + Data_name)
    return player_list
```

- Thu thập trang dữ liệu cầu thủ từ 10 trang web:

2023-2024 Premier League Player Stats | FBref.com
2023-2024 Premier League Goalkeeper Stats | FBref.com
2023-2024 Premier League Shooting Stats | FBref.com
2023-2024 Premier League Passing Stats | FBref.com
2023-2024 Premier League Passing Stats | FBref.com
2023-2024 Premier League Goal and Shot Creation | FBref.com
2023-2024 Premier League Defensive Action Stats | FBref.com
2023-2024 Premier League Possession Stats | FBref.com
2023-2024 Premier League Playing Time | FBref.com
2023-2024 Premier League Miscellaneous Stats | FBref.com

Lấy dữ liệu ở bảng Player Standard như hình ảnh, tương tự với 9 trang web còn lại



- Tạo tiêu đề cho bảng kết quả, gồm có 172 chỉ số (tính cả cột Name), tiêu đề gồm 3 hàng:

```
column_titles = pd.MultiIndex.from_tuples([
    ('', '', 'Name'), ('', '', 'Nation'), ('', '', 'Team'), ('', '', 'Position'), ('', '', 'Age'),
    ('', '', 'Matches Played'),
    ('', 'Playing Time', 'Starts'), ('', 'Playing Time', 'Min'),
    ('', 'Performance', 'non-Penalty Goals'), ('', 'Performance', 'Penalty Goals'),('', 'Performance', 'Assists'), ('', 'Performance', ' Yellow Cards'), ('', 'Performance', 'Red Cards')
    ('', 'Expected', 'xG'), ('', 'Expected', 'npxG'), ('', 'Expected', 'xAG'),
    ('', 'Progression', 'PrgC'), ('', 'Progression', 'PrgP'), ('', 'Progression', 'PrgR'),
    ('', 'Per 90 minutes', 'Gls'), ('', 'Per 90 minutes', 'Ast'), ('', 'Per 90 minutes', 'G+A'),
    ('', 'Per 90 minutes', 'G-PK'), ('', 'Per 90 minutes', 'G+A-PK'), ('', 'Per 90 minutes', 'xG'),
    ('', 'Per 90 minutes', 'xAG'), ('', 'Per 90 minutes', 'xG+xAG'), ('', 'Per 90 minutes', 'npxG'),('', 'Per 90 minutes','npxG + xAG'),
    ('Goalkeeping', 'Performance', 'GA'), ('Goalkeeping', 'Performance', 'GA90'),
    ('Goalkeeping', 'Performance', 'SoTA'), ('Goalkeeping', 'Performance', 'Saves'),
    ('Goalkeeping', 'Performance', 'Save%'), ('Goalkeeping', 'Performance', 'W'),
    ('Goalkeeping', 'Performance', 'D'), ('Goalkeeping', 'Performance', 'L'),
    ('Goalkeeping', 'Performance', 'CS'), ('Goalkeeping', 'Performance', 'CS%'),
    ('Goalkeeping', 'Penalty Kick', 'PKatt'), ('Goalkeeping', 'Penalty Kick', 'PKA'),
    ('Goalkeeping', 'Penalty Kick', 'PKsv'), ('Goalkeeping', 'Penalty Kick', 'PKm'),
    ('Goalkeeping', 'Penalty Kick', 'Save%'),
    ('Shooting', 'Standard', 'Gls'), ('Shooting', 'Standard', 'Sh'), ('Shooting', 'Standard', 'SoT'),
    ('Shooting', 'Standard', 'SoT%'), ('Shooting', 'Standard', 'Sh/90'), ('Shooting', 'Standard', 'SoT/90'),
    ('Shooting', 'Standard', 'G/Sh'), ('Shooting', 'Standard', 'G/SoT'), ('Shooting', 'Standard', 'Dist'),
    ('Shooting', 'Standard', 'FK'), ('Shooting', 'Standard', 'PK'), ('Shooting', 'Standard', 'PKatt'),
    ('Shooting', 'Expected', 'xG'), ('Shooting', 'Expected', 'npxG'), ('Shooting', 'Expected', 'npxG/Sh'),
    ('Shooting', 'Expected', 'G-xG'), ('Shooting', 'Expected', 'np:G-xG'),
    ('Passing', 'Total', 'Cmp'), ('Passing', 'Total', 'Att'), ('Passing', 'Total', 'Cmp%'),
    ('Passing', 'Total', 'TotDist'), ('Passing', 'Total', 'PrgDist'),
    ('Passing', 'Short', 'Cmp'), ('Passing', 'Short', 'Att'), ('Passing', 'Short', 'Cmp%'),
    ('Passing', 'Medium', 'Cmp'), ('Passing', 'Medium', 'Att'), ('Passing', 'Medium', 'Cmp%'),
    ('Passing', 'Long', 'Cmp'), ('Passing', 'Long', 'Att'), ('Passing', 'Long', 'Cmp%'),
    ('Passing', 'Expected', 'Ast'), ('Passing', 'Expected', 'xAG'), ('Passing', 'Expected', 'xA'),
    ('Passing', 'Expected', 'A-xAG'), ('Passing', 'Expected', 'KP'), ('Passing', 'Expected', '1/3'),
    ('Passing', 'Expected', 'PPA'), ('Passing', 'Expected', 'CrsPA'), ('Passing', 'Expected', 'PrgP'),
    ('Pass Types', 'Pass Types', 'Live'), ('Pass Types', 'Pass Types', 'Dead'), ('Pass Types', 'Pass Types', 'FK'),
    ('Pass Types', 'Pass Types', 'TB'), ('Pass Types', 'Pass Types', 'Sw'), ('Pass Types', 'Pass Types', 'Crs'),
    ('Pass Types', 'Pass Types', 'TI'), ('Pass Types', 'Pass Types', 'CK'),
    ('Pass Types', 'Corner Kicks', 'In'), ('Pass Types', 'Corner Kicks', 'Out'), ('Pass Types', 'Corner Kicks', 'Str'),
    ('Pass Types', 'Outcomes', 'Cmp'), ('Pass Types', 'Outcomes', 'Off'), ('Pass Types', 'Outcomes', 'Blocks'),
    ('Goal and Shot Creation', 'SCA', 'SCA'), ('Goal and Shot Creation', 'SCA', 'SCA90'),
    ('Goal and Shot Creation', 'SCA Types', 'PassLive'), ('Goal and Shot Creation', 'SCA Types', 'PassDead'),
    ('Goal and Shot Creation', 'SCA Types', 'TO'), ('Goal and Shot Creation', 'SCA Types', 'Sh'),
    ('Goal and Shot Creation', 'SCA Types', 'Fld'), ('Goal and Shot Creation', 'SCA Types', 'Def'),
    ('Goal and Shot Creation', 'GCA', 'GCA'), ('Goal and Shot Creation', 'GCA', 'GCA90'),
    ('Goal and Shot Creation', 'GCA Types', 'PassLive'), ('Goal and Shot Creation', 'GCA Types', 'PassDead'),
```

- Lần lượt lấy dữ liệu từ các danh sách đã liệt kê, gọi hàm GetDataFromWeb(url, Xpath_player, Data_name)

Ví dụ: Lấy dữ liệu từ trang web: [2023-2024 Premier League Player Stats | FBref.com]

Phân tích mã HTML tìm các bảng theo biến Xpath_player với

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

+ Xpath_player = '//*[@id="stats_standard"]'

+ Data_name = "Standard"

- Hàm trả về 1 danh sách các người chơi kèm chỉ số, tạo 1 dataframe để lữu trữ

- Tương tự cho 9 bảng còn lại. 10 bảng đều chọn cột Name, và Team làm khóa

```
# Lấy dữ liệu từ bảng stats standard
url = "https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats"
Xpath_player = '//*[@id="stats_standard"]'
Data_name = "Standard"
list = GetDataFromWeb(url, Xpath_player, Data_name)

player_list = []
for p in list:
    try:
        Name, Nation, Position, Team, Age = p[0:5]
        mp, starts, min = p[6:9]
        non_pen, pen_goal, ass, ycard, rcard = [p[13],p[14],p[11],p[16],p[17]]
        xG, npxG, xAG = p[18:21]
        PrgC, PrgP, PrgR = p[22:25]
        Gls, Ast, G_Ao, G_PKo, G_A_PKo, xG, xAG, xG_xAG, npxG, npxG_xAG = p[25:35]
        if min > 90: player_list.append([Name, Nation, Team, Position, Age,mp, starts, min,non_pen, pen_goal, ass, ycard, rcard,xG, npxG, xAG,PrgC, PrgP, PrgR,Gls, Ast, G_Ao, G_PKo, G_A
    except IndexError:
        break
df_player = pd.DataFrame(player_list, columns=['Name','Nation','Team','Position','Age', 'Matches Played','Starts', 'Min', 'non-Penalty Goals', 'Penalty Goals', 'Assists', ' Yellow Cards
                                               'xG', 'npxG', 'xAG',

                                'PrgC', 'PrgP', 'PrgR',
                                'Gls', 'Ast', 'G+A', 'G-PK', 'G+A-PK', 'xG', 'xAG', 'xG+xAG', 'npxG','npxG + xAG'])
# df_player.to_csv('Standard.csv')
################################################################################
#Lấy dữ liệu từ bảng player keepers
url = "https://fbref.com/en/comps/9/2023-2024/keepers/2023-2024-Premier-League-Stats"
Xpath_player = '//*[@id="stats_keeper"]'
Data_name = "Keepers"
list = GetDataFromWeb(url, Xpath_player, Data_name)
keeper_list = []

for p in list:
    Name = p[0]
    Team = p[3]
    GAo, GA90o, SoTA, Saves, Save__o, Wo, Do, Lo, CSo, CS_o, PKatt, PKAo, PKsv, PKm, Save_o  = p[10:]
    keeper_list.append([Name,Team,GAo, GA90o, SoTA, Saves, Save__o, Wo, Do, Lo, CSo, CS_o, PKatt, PKAo, PKsv, PKm, Save_o])
df_keepers = pd.DataFrame(keeper_list, columns=['Name','Team','GA', 'GA90', 'SoTA', 'Saves', 'Save%', 'W', 'D', 'L', 'CS','CS%', 'PKatt', 'PKA', 'PKsv', 'PKm', 'Save%'] )
```

- Sau khi lấy được hết các dữ liệu từ 10 bảng ở trên, thực hiên ghép 10 dataframe lại làm dựa trên cột Name và Team. Sau khi ghép ta có một datafram chứa dữ liệu cầu thủ gồm đầy đủ 172 chỉ số.
- Tạo thêm 1 cột First Name để sắp xếp
- Sử dụng hàm sorted_values theo First Name và Age để sắp xếp các cầu thủ trong bảng theo: Thứ tự các cầu thủ sắp xếp theo thứ tự tên (First Name), nếu trùng tên thì xếp theo độ tuổi từ lớn đến nhỏ.
- Sau khi xắp xếp xong, xóa cột First Name
- Ghép colum_titles đã tạo ở trên để tạo bảng với 3 dòng tiêu đề
- Lưu bảng vào file 'result.csv' , điền giá trị không có, không áp dụng là 'N/a'

```
# Gộp các dataframe lại thành 1
dataframes = [df_keepers,df_shooting,df_Passing, df_passing_type,df_Goal_and_Shot_Creation,df_Defensive_Actions,df_possession, df_playing_time,df_Miscellaneous_Stats]
df_merged = df_player
for df in dataframes:
    df_merged = pd.merge(df_merged,df, on = ['Name','Team'], how = 'left')
+################################################################################
# Sắp xếp theo First Name và Age
df_merged['First Name'] = df_merged['Name'].apply(lambda x: x.split()[0])
df_sorted = df_merged.sort_values(by=['Name', 'Age'], ascending=[True, False])
df_sorted = df_sorted.drop(columns=['First Name'])
# In ra DataFrame đã sắp xếp
df_sorted.columns = column_titles
df_sorted.to_csv('result.csv', index=False, na_rep='N/a')
print('Du lieu da duoc luu vao file result.csv')
```

IV. Kết quả và phân tích

- Kết quả được lưu trong file 'result.csv'
- Bảng kết quả có 493 cầu thủ, mỗi cầu thủ thu thập được 172 chỉ số

B. Bài 2

I. Phân tích bài toán

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

Sử dụng phương thức nlargest và nsmallest: Sử dụng các phương thức này để tìm ra 3 cầu thủ có điểm số cao nhất và thấp nhất cho từng chỉ số.

- Tìm trung vị, trung bình và độ lệch chuẩn:

Trung vị: Sử dụng phương thức median() của pandas.

Trung bình: Sử dụng phương thức mean() của pandas.

Độ lệch chuẩn: Sử dụng phương thức std() của pandas.

Tính toán cho toàn giải: Tính toán cho tất cả cầu thủ trong giải.

Tính toán cho từng đội: Tính toán các thống kê trên cho từng đội bóng.

- **Sử dụng matplotlib**: Sử dụng thư viện matplotlib để vẽ histogram cho từng chỉ số.
- So sánh các chỉ số của các đội: Xác định đội bóng nào có điểm số cao nhất cho từng chỉ số.
- Đánh giá phong độ: Dựa trên các chỉ số tổng thể, đánh giá đội nào có phong độ tốt nhất trong giải Ngoại Hạng Anh mùa 2023-2024.

II. Cài đặt và công cụ

Các thư viện sử dụng trong bài 2:

- matplotlib.pyplot
- seaborn
- pandas

III. Code

1. Tính trung vị, trung bình, độ lệch chuẩn

- Dữ liệu được đọc từ tệp result.csv, với các hàng đầu tiên (header) có cấu trúc đa cấp (3 cấp), có ba hàng đầu làm tên cột.
- Một số cột có tên bắt đầu với "Unnamed:" (kết quả khi tệp CSV có các ô trống ở hàng đầu tiên). Để xử lý điều này, mã sẽ đặt tên các cột này thành chuỗi rỗng '', nhằm làm sạch tên cột.
- Biến final_results là danh sách chứa kết quả tính toán cuối cùng, với mỗi hàng là kết quả của một đội hoặc toàn bộ giải.
- Vòng lặp for duyệt qua các cột từ cột thứ 4 (chỉ số col=4) để bỏ qua các thông tin ban đầu
- Với mỗi cột:

- **Toàn giải:** Tính median, mean và std cho toàn bộ người chơi trong giải và thêm vào final_results.
- **Theo đội:** Sử dụng groupby để tính median, mean và std cho từng đội

- Với từng đội, mã sẽ kiểm tra xem đội đó đã có mặt trong final_results chưa:
  - Nếu đội đã có, các chỉ số mới sẽ được thêm vào hàng tương ứng.
  - Nếu đội chưa có, một hàng mới sẽ được tạo cho đội đó và thêm vào final_results.

Bai2_Mean_Media_Std.py > ...

```python
import pandas as pd
# Đọc tệp CSV với header đa tầng từ file 'result.csv'
df = pd.read_csv('result.csv', header=[0, 1, 2])
# Làm sạch tên cột
df.columns = [tuple('' if 'Unnamed:' in x else x for x in col) for col in df.columns]
# Tạo một DataFrame để lưu kết quả cuối cùng
final_results = []
# Duyệt qua các attribute (bắt đầu từ cột thứ 6 trong bảng)
for col in range(4, df.shape[1]):
    column = df.iloc[:, col]
    column_numeric = pd.to_numeric(column, errors='coerce')  # Chuyển cột về dạng số

    # Tính toán median, mean, và std cho toàn giải
    median_value_all = round(column_numeric.median(), 2)
    mean_value_all = round(column_numeric.mean(), 2)
    std_value_all = round(column_numeric.std(), 2)

    # Thêm kết quả cho toàn giải
    if col == 4:  # Nếu là attribute đầu tiên
        final_results.append(['All', median_value_all, mean_value_all, std_value_all])
    else:  # Các attribute khác
        final_results[0].extend([median_value_all, mean_value_all, std_value_all])

    # Tính toán cho từng đội
    for team_name, team_data in df.groupby(('', '', 'Team')):
        team_column = pd.to_numeric(team_data.iloc[:, col], errors='coerce')  # Cột của từng đội
        team_median = round(team_column.median(), 2)
        team_mean = round(team_column.mean(), 2)
        team_std = round(team_column.std(), 2)

        # Kiểm tra nếu đội đã có trong final_results
        found = False
        for row in final_results:
            if row[0] == team_name:
                # Cập nhật hàng của đội với các chỉ số mới
                row.extend([team_median, team_mean, team_std])
                found = True
                break
        if not found:
            # Nếu đội chưa có trong final_results, thêm mới
            final_results.append([team_name, team_median, team_mean, team_std])
```

- Sau khi tính toán xong, final_results sẽ được chuyển đổi thành một DataFrame (final_df) để dễ dàng xử lý và ghi ra tệp CSV.
- Kết quả cuối cùng sẽ được in ra màn hình và ghi vào tệp result2.csv với các chỉ số thống kê cho toàn bộ giải đấu và từng đội.

```python
# Chuyển đổi danh sách kết quả cuối cùng thành DataFrame
final_df = pd.DataFrame(final_results)

# Cập nhật tiêu đề cột để bao gồm tên chỉ số cho mỗi attribute
column_titles = ['Team']
for col_index in range(4, df.shape[1]):
    att = "_".join(filter(None, df.columns[col_index]))  # Kết hợp tên cột
    column_titles.extend([f'Median of {att}', f'Mean of {att}', f'Std of {att}'])

final_df.columns = column_titles

# Ghi kết quả ra file results2.csv
print(final_df)
final_df.to_csv('result2.csv', index=False)
```

2. Vẽ biểu đồ Histogram

- Dữ liệu được đọc từ tệp result.csv với cấu trúc header đa cấp (header=[0,1,2]), tức là tệp có ba hàng đầu làm tên cột.
- Một số cột có tên bắt đầu với "Unnamed:" (kết quả khi tệp CSV có các ô trống ở hàng đầu tiên). Để xử lý điều này, mã sẽ đặt tên các cột này thành chuỗi rỗng '', nhằm làm sạch tên cột.
- Dựa vào kiểu dữ liệu (float và int), mã sẽ lọc các cột có giá trị số (numeric) và lưu vào biến numeric_cols. Các cột này sẽ được dùng để vẽ biểu đồ.
- Tạo thư mục hinh_ve để lưu hình ảnh histogram
- Tạo hàm làm sạch các kí tự k hợp lệ khỏi tên tệtệp

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import os
import re
# Đọc dữ liệu từ CSV với header đa cấp
df = pd.read_csv('result.csv', header=[0, 1, 2])

# Làm sạch tên cột
df.columns = [tuple('' if 'Unnamed:' in x else x for x in col) for col in df.columns]

# Lọc các cột có kiểu dữ liệu số (float và int)
numeric_cols = df.select_dtypes(include=['float', 'int']).columns
# Đường dẫn đến thư mục
output_dir = 'hinh_ve'

# Tạo thư mục nếu nó không tồn tại
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
#Tắt chế độ tương tác
plt.ioff()
# Hàm để làm sạch tên tệp
def clean_filename(name):
    # Thay thế các ký tự không hợp lệ bằng dấu gạch dưới
    return re.sub(r'[<>:"/\\|?*]', '_', name)
```

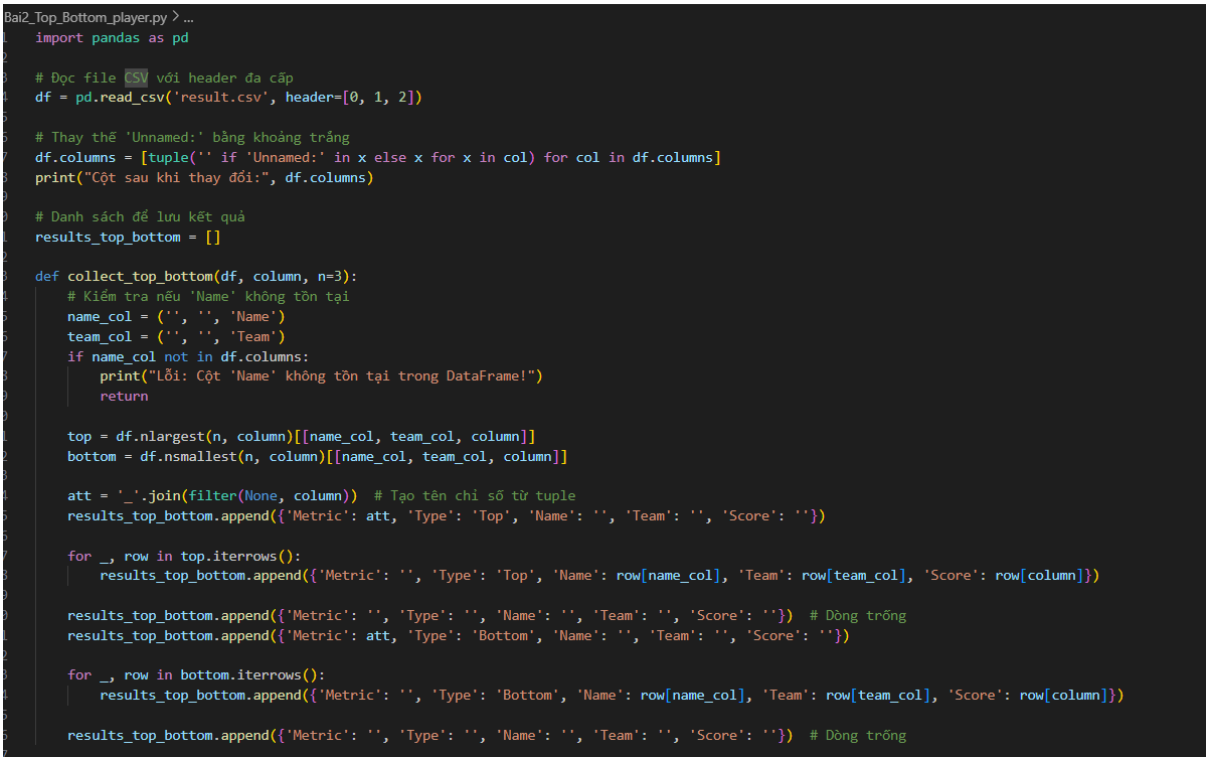
- Đối với mỗi thuộc tính số trong numeric_cols, mã thực hiện hai thao tác:
    - **Vẽ biểu đồ toàn bộ:** Vẽ biểu đồ histogram cho toàn bộ người chơi trong dữ liệu với trục x là giá trị của thuộc tính và trục y là tần số (số lượng).
    - **Vẽ biểu đồ theo từng đội:** Sử dụng FacetGrid để tạo biểu đồ histogram cho từng đội, giúp người xem dễ dàng so sánh phân phối của thuộc tính này giữa các đội khác nhau.
- Sử dụng tên cột của thuộc tính để tạo tiêu đề và nhãn trục x cho biểu đồ. Tên thuộc tính (dạng tuple) được ghép lại thành chuỗi his_name để hiển thị rõ ràng.

```python
# Vẽ biểu đồ histogram cho từng thuộc tính số theo từng đội
for cnt, att in enumerate(numeric_cols, start=1):
    # Ghép tên cột để tạo tiêu đề cho biểu đồ
    his_name = "_".join(filter(None, att))  # Kết hợp các phần trong tuple không rỗng
    his_name_cleaned = clean_filename(his_name)  # Làm sạch tên tệp
  # Kết hợp các phần trong tuple không rỗng

    # Biểu đồ toàn bộ dữ liệu
    plt.figure(figsize=(10, 6))
    sns.histplot(data=df, x=att, kde=True, bins=20)
    plt.title(f"Histogram {his_name} of All Players in the League")
    his_name_cleaned = clean_filename(his_name)  # Làm sạch tên tệp
    plt.xlabel(his_name)
    plt.ylabel("Tần suất")

    # Lưu biểu đồ vào thư mục hinh ve
    plt.savefig(os.path.join(output_dir, f'histogram_{his_name_cleaned}.png'))
    plt.close()   # Đóng hình để tránh hiển thị

    # Biểu đồ theo từng đội
    g = sns.FacetGrid(df, col=('','','Team'), col_wrap=4, height=3, sharex=True, sharey=True)
    g.map_dataframe(sns.histplot, x=att, bins=20, kde=True)
    g.set_axis_labels(his_name, "Tần suất")  # Đặt tên trục x
    g.set_titles(f"{his_name} of {{col_name}}")  # Tiêu đề

    # Căn chỉnh và hiển thị biểu đồ
    plt.tight_layout()
    plt.savefig(os.path.join(output_dir, f'facet_{his_name_cleaned}.png'))
    plt.close()  # Đóng hình để tránh hiển thị
```

-Hình ảnh thu được tự động lưu vào thư mục hinh_ve

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

- Sử dụng header=[0, 1, 2] để đọc dữ liệu với ba hàng đầu tiên làm tên cột, mỗi cột được đại diện bởi một bộ ba (tuple) với tên từ các hàng đó.
- Các tên cột không có giá trị (một số hàng trong header có ô trống và hiển thị là Unnamed:) được thay thế bằng chuỗi rỗng ''.
- results_top_bottom là một danh sách trống để lưu kết quả top và bottom cho từng chỉ số.
- Hàm collect_top_bottom nhận vào DataFrame df, tên cột column, và số lượng hàng n để chọn ra n hàng có giá trị cao nhất (top) và n hàng có giá trị thấp nhất (bottom).
- Kiểm tra nếu cột ('','','Name') không tồn tại để đảm bảo có thể lấy được tên người chơi.
- Duyệt cột và ghi tên chỉ số (attribute): Sử dụng vòng lặp for để lấy tên cột và tạo tên chỉ số dạng att.
- Thêm kết quả vào results_top_bottom:
    - Thêm hàng chứa tên chỉ số và đánh dấu là "Top" cho các giá trị cao nhất.
    - Lặp qua các hàng trong top và bottom để thêm từng hàng với tên người chơi (Name), đội (Team), và điểm số (Score).
    - Thêm dòng trống giữa các nhóm để dễ phân biệt khi xuất ra file.



```python
import pandas as pd

# Đọc file CSV với header đa cấp
df = pd.read_csv('result.csv', header=[0, 1, 2])

# Thay thế 'Unnamed:' bằng khoảng trắng
df.columns = [tuple('' if 'Unnamed:' in x else x for x in col) for col in df.columns]
print("Cột sau khi thay đổi:", df.columns)

# Danh sách để lưu kết quả
results_top_bottom = []

def collect_top_bottom(df, column, n=3):
    # Kiểm tra nếu 'Name' không tồn tại
    name_col = ('', '', 'Name')
    team_col = ('', '', 'Team')
    if name_col not in df.columns:
        print("Lỗi: Cột 'Name' không tồn tại trong DataFrame!")
        return

    top = df.nlargest(n, column)[[name_col, team_col, column]]
    bottom = df.nsmallest(n, column)[[name_col, team_col, column]]

    att = '_'.join(filter(None, column))  # Tạo tên chỉ số từ tuple
    results_top_bottom.append({'Metric': att, 'Type': 'Top', 'Name': '', 'Team': '', 'Score': ''})

    for _, row in top.iterrows():
        results_top_bottom.append({'Metric': '', 'Type': 'Top', 'Name': row[name_col], 'Team': row[team_col], 'Score': row[column]})

    results_top_bottom.append({'Metric': '', 'Type': '', 'Name': '', 'Team': '', 'Score': ''})  # Dòng trống
    results_top_bottom.append({'Metric': att, 'Type': 'Bottom', 'Name': '', 'Team': '', 'Score': ''})

    for _, row in bottom.iterrows():
        results_top_bottom.append({'Metric': '', 'Type': 'Bottom', 'Name': row[name_col], 'Team': row[team_col], 'Score': row[column]})

    results_top_bottom.append({'Metric': '', 'Type': '', 'Name': '', 'Team': '', 'Score': ''})  # Dòng trống
```

- Duyệt qua các cột số và thu thập top/bottom:

- Duyệt qua tất cả các cột số từ cột thứ 6 (df.columns[5:]), vì có thể các cột đầu tiên chứa thông tin không phải số như tên hoặc mã ID.
- Gọi hàm collect_top_bottom cho từng cột để thêm kết quả top/bottom vào results_top_bottom.

- Sau khi thu thập xong, danh sách results_top_bottom được chuyển thành DataFrame (results_df) để dễ dàng ghi ra file.
- Kết quả được lưu vào tệp top_bottom.txt dưới dạng văn bản và top_bottom.csv dưới dạng CSV.

```python
# Duyệt qua các chỉ số dạng số và tìm top/bottom
numeric_cols = df.columns[5:]  # Lấy cột từ thứ 6 trở đi
for col in numeric_cols:
    if pd.api.types.is_numeric_dtype(df[col]):
        collect_top_bottom(df, col)

# Chuyển danh sách kết quả thành DataFrame
results_df = pd.DataFrame(results_top_bottom)

# Ghi vào file txt
with open('top_bottom.txt', 'w', encoding='utf-8') as file:
    file.write(results_df.to_string(index=False))  # Không ghi chỉ số

# Ghi kết quả ra file CSV
results_df.to_csv('top_bottom.csv', index=False)
print("Danh sách đã được lưu vào file top_bottom.csv")
```

4. Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số

- Đọc dữ liệu từ file CSV có tên result2.csv và lưu vào biến df dưới dạng DataFrame.

- top_teams_by_mean = []: Tạo một danh sách rỗng để lưu kết quả.
- Lặp qua từng cột trong DataFrame, Kiểm tra xem tên cột có bắt đầu bằng "Mean of" hay không. Điều này có nghĩa là cột này chứa giá trị trung bình của một chỉ số nào đó.
- Sử dụng idxmax() để tìm chỉ số của hàng có giá trị trung bình cao nhất trong cột hiện tại và lấy tên đội từ cột "Team".
- Lấy giá trị tối đa (điểm trung bình cao nhất) từ cột hiện tại.
- Thêm thông tin về chỉ số, tên đội và giá trị điểm trung bình cao nhất vào danh sách top_teams_by_mean.
- Chuyển danh sách top_teams_by_mean thành một DataFrame mới với các cột được đặt tên là 'Attribute', 'Top Team', và 'Mean Value'.
- In ra DataFrame để kiểm tra kết quả.
- Ghi DataFrame vào file CSV có tên top_teams_by_mean.csv

```python
# ai2_top_teams.py > ...
import pandas as pd

# Đọc dữ liệu từ file CSV
df = pd.read_csv("result2.csv")

# Tìm đội có điểm cao nhất ở mỗi chỉ số dựa trên giá trị trung bình (mean)
top_teams_by_mean = []

# Lặp qua từng cột để tìm đội bóng có điểm trung bình cao nhất
for column in df.columns:
    if column.startswith('Mean of'):
        max_mean_team = df.loc[df[column].idxmax(), 'Team']  # Tên đội bóng
        max_mean_value = df[column].max()  # Giá trị điểm trung bình cao nhất
        top_teams_by_mean.append([column.replace('Mean of ', ''), max_mean_team, max_mean_value])

# Tạo DataFrame từ danh sách kết quả
top_teams_df = pd.DataFrame(top_teams_by_mean, columns=['Attribute', 'Top Team', 'Mean Value'])
print(top_teams_df)

# Ghi kết quả vào file CSV
top_teams_df.to_csv('top_teams_by_mean.csv', index=False)
```

* Tìm đội có phong độ tốt nhất
- Tạo danh sách các cột chứa giá trị trung bình.
- Tính tổng điểm trung bình cho mỗi đội bằng cách cộng tất cả các giá trị trung bình của các cột được chọn.
- Tìm tên đội có tổng điểm trung bình cao nhất.
- Lấy giá trị tổng điểm cao nhất.
- In ra tên đội bóng có phong độ tốt nhất và tổng điểm trung bình của họ

```python
# Tìm đội có phong độ tốt nhất
mean_columns = [col for col in df.columns if col.startswith('Mean of')]
df['Total Mean Score'] = df[mean_columns].sum(axis=1)  # Tính tổng điểm trung bình

# Tìm đội bóng có phong độ tốt nhất
top_team = df.loc[df['Total Mean Score'].idxmax(), 'Team']
top_score = df['Total Mean Score'].max()
print(f"Đội bóng có phong độ tốt nhất: {top_team} với tổng điểm trung bình: {top_score}")
```

- Đội bóng có phong độ tốt nhất là Manchester City

```
PS C:\Users\lecao\Documents\python\BTL> & C:/Users/lecao/AppData/Local/Programs/Python/Python312/python.exe c:/Users/lecao/Documents/python/BTL/Bai2_Top_teams.py
                              Attribute         Top Team  Mean Value
0                                   Age         West Ham       28.27
1                         Matches Played           Fulham       27.24
2                    Playing Time_Starts           Fulham       19.90
3                       Playing Time_Min  Manchester City     1785.57
4             Performance_non-Penalty Goals  Manchester City        4.05
..                                    ...              ...         ...
163     Miscellaneous Stats_Performance_OG    Sheffield Utd        0.23
164  Miscellaneous Stats_Performance_Recov        Liverpool       92.00
165   Miscellaneous Stats_Aerial Duels_Won          Everton       29.39
166  Miscellaneous Stats_Aerial Duels_Lost      Bournemouth       25.65
167  Miscellaneous Stats_Aerial Duels_Won%  Nott'ham Forest       55.80

[168 rows x 3 columns]
Đội bóng có phong độ tốt nhất: Manchester City với tổng điểm trung bình: 50447.44000000002
PS C:\Users\lecao\Documents\python\BTL>
```

C. Bài 3

I. Phân tích bài toán

Mục tiêu chính của bài toán này là phân loại các cầu thủ thành các nhóm có đặc điểm giống nhau dựa trên các chỉ số

Phân tích bài toán của bạn bao gồm một số bước phân tích và trực quan hóa dữ liệu cầu thủ, cụ thể là:

1. Bài toán phân cụm cầu thủ bằng thuật toán K-means

Mục tiêu chính của bài toán này là phân loại các cầu thủ thành các nhóm có đặc điểm giống nhau dựa trên các chỉ số như tuổi, số trận đã chơi, số bàn thắng, số bàn thắng kỳ vọng (xG), số lần sút trúng đích, v.v.

- Dữ liệu đầu vào: Một bảng dữ liệu chứa thông tin chi tiết của các cầu thủ, bao gồm các chỉ số về phong độ, hiệu suất và thống kê thi đấu.
- Dữ liệu đầu ra: Các nhóm cầu thủ có đặc điểm tương đồng, với mỗi nhóm đại diện cho một loại cầu thủ với phong cách hoặc vai trò tương tự.

Mục đích: Việc phân cụm các cầu thủ giúp hiểu rõ hơn về đặc điểm và vai trò của từng cầu thủ dựa trên dữ liệu thống kê, từ đó hỗ trợ phân tích và đánh giá đội hình.

2. Xác định số lượng nhóm phù hợp cho K-means

Để phân loại các cầu thủ thành các nhóm có đặc điểm giống nhau, cần xác định số lượng nhóm (cụm) tối ưu cho K-means.

- **Phương pháp Elbow**: Để chọn số lượng cụm, ta có thể sử dụng phương pháp Elbow dựa trên giá trị SSE (Sum of Squared Errors). Mục tiêu là chọn một điểm uốn trên biểu đồ, nơi mà sự giảm SSE bắt đầu chững lại khi tăng số cụm. Đây là dấu hiệu cho thấy việc thêm cụm không còn cải thiện phân cụm nhiều nữa.
- **Phương pháp Silhouette Score**: Điểm Silhouette đo lường mức độ tương tự của các điểm dữ liệu trong cùng cụm và khác cụm. Giá trị cao cho thấy cụm được phân biệt rõ ràng. Chọn số cụm với điểm Silhouette cao giúp tối ưu hóa sự gắn kết và tách biệt của các cụm.

Sau khi chọn số cụm, ta có thể sử dụng PCA để giảm số chiều dữ liệu xuống 2 chiều để dễ dàng trực quan hóa.

PCA (Principal Component Analysis) là phương pháp giảm số chiều phổ biến, giúp giữ lại phần lớn thông tin quan trọng trong dữ liệu gốc.

Vẽ biểu đồ phân cụm: Sau khi giảm số chiều, chúng ta sẽ sử dụng K-means để phân cụm trên dữ liệu 2D. Việc trực quan hóa phân cụm trên mặt phẳng 2D giúp ta dễ dàng nhận diện các nhóm cầu thủ, từ đó thấy rõ cách các cầu thủ được phân loại dựa trên các chỉ số thống kê.

3. Vẽ biểu đồ radar

Biểu đồ radar là công cụ hữu ích để so sánh chi tiết từng chỉ số của hai cầu thủ. Điều này đặc biệt hữu ích khi muốn so sánh các cầu thủ cùng vị trí hoặc vai trò trong đội hình.

Dữ liệu đầu vào: Tên hai cầu thủ và danh sách các chỉ số cần so sánh.

Dữ liệu đầu ra: Biểu đồ radar, trong đó mỗi trục đại diện cho một chỉ số. Các giá trị của hai cầu thủ sẽ được hiển thị trên cùng biểu đồ để dễ dàng so sánh.
Ứng dụng: Biểu đồ này giúp ta có cái nhìn trực quan về sự khác biệt và tương đồng giữa hai cầu thủ trong các khía cạnh như độ tuổi, số trận đã chơi, số bàn thắng, xG, và số lần sút trúng đích.

II. Cài đặt

- numpy
- seaborn
- matplotlib
- sklearn

III. Code

1. Thuật toán K-means để phân loại các cầu thủ

- Import các thư viện cần thiết
- Đọc dữ liệu từ file result.csv với nhiều header. Sau đó, xử lý tên cột để loại bỏ các phần không cần thiết.
- In ra thông tin cơ bản về DataFrame, kiểm tra số lượng giá trị thiếu trong mỗi cột, và đếm số lượng đội bóng trong cột 'Team'.

Bai3.py > ...

```python
import warnings
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA

warnings.filterwarnings('ignore')

# Đọc vào file result.csv
df = pd.read_csv('result.csv', header=[0, 1, 2])
df.columns = [tuple('' if 'Unnamed:' in x else x for x in col) for col in df.columns]

# Kiểm tra thông tin cơ bản của DataFrame
print("DataFrame Info:")
df.info()

print("\nMissing Values per Column:")
print(df.isna().sum())

print("\nTeam Value Counts:")
print(df[('', '', 'Team')].value_counts())
```

- Lặp qua một khoảng giá trị số cụm (từ 2 đến 9) để phù hợp với mô hình K-means.
- Đối với mỗi giá trị k, thêm giá trị SSE (inertia) và điểm silhouette vào danh sách tương ứng.

```
# K-means Clustering
sse = []
silhouette_scores = []
# Lọc dữ liệu dạng số để phân cụm
X = df.select_dtypes(include=[float, int]).dropna(axis=1, how='any')  # Xóa các cột có giá trị null

for k in range(2, 10):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    sse.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X, kmeans.labels_))
```

- Tạo hai biểu đồ con: một cho Phương pháp Elbow (SSE so với số lượng cụm) và một cho điểm silhouette.

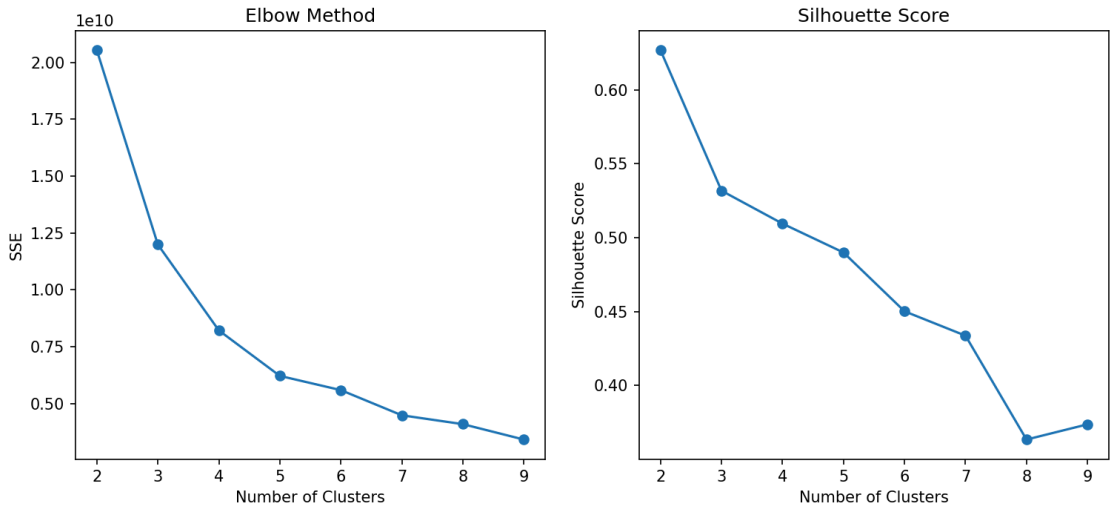
```
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(range(2, 10), sse, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title("Elbow Method")

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()
```

+ Vẽ biểu đồ SSE (Tổng bình phương sai số) theo số lượng cụm (k). Khi số cụm tăng lên, SSE thường giảm vì dữ liệu được phân chia tốt hơn. Tuy nhiên, sau một số lượng cụm nhất định, sự giảm sút này sẽ chậm lại, tạo thành một "khuỷu". Điểm khuỷu này được coi là số cụm tối ưu.

+ Silhouette score đo lường mức độ gần gũi giữa các đối tượng trong cùng một cụm và độ xa cách giữa các cụm khác nhau. Điểm này nằm trong khoảng từ -1 đến 1, với giá trị cao hơn cho thấy cụm tốt hơn.

+ Từ biểu đồ khuỷu, bạn tìm số cụm tại điểm khuỷu để xác định số cụm tối ưu.

+ Từ biểu đồ silhouette, chọn số cụm có điểm silhouette cao nhất.



- **Phương pháp Elbow (SSE)**: Biểu đồ bên trái cho thấy khi tăng số cụm, SSE (Sum of Squared Errors) giảm dần. Điểm gấp khúc rõ nhất (elbow point) ở đây có vẻ là khi số cụm là 4, vì từ 4 trở đi, SSE giảm chậm lại hơn.

- **Phương pháp Silhouette Score**: Biểu đồ bên phải cho thấy Silhouette Score giảm dần khi số cụm tăng. Điểm số cao nhất là khi số cụm là 2, tuy nhiên số cụm 4 vẫn có giá trị tương đối cao so với các cụm lớn hơn.

=> Nên phân loại cầu thủ thành 4 nhóm (Số cluster = 4)

2. Thuật toán PCA - Giảm dữ liệu xuống 2 chiều

- Tạo một đối tượng PCA từ thư viện sklearn.decomposition. PCA (Principal Component Analysis) là một kỹ thuật giảm chiều dữ liệu, giúp chúng ta biểu diễn dữ liệu trong không gian với ít chiều hơn nhưng vẫn giữ được nhiều thông tin quan trọng.. Ở đây, n_components=2 chỉ định rằng chúng ta muốn giảm dữ liệu xuống còn 2 chiều (hai thành phần chính), để dễ dàng trực quan hóa trên biểu đồ 2D.
- Dữ liệu gốc X (chỉ bao gồm các biến số) được biến đổi bằng phương pháp PCA. fit_transform thực hiện việc tìm các thành phần chính từ dữ liệu gốc và sau đó chuyển đổi dữ liệu sang không gian của các thành phần chính đó. Kết quả là X_pca, một mảng hai chiều (2D) của dữ liệu đã được giảm chiều.
- Tạo một đối tượng KMeans để thực hiện phân cụm. Ở đây, n_clusters=4 chỉ định rằng chúng ta muốn chia dữ liệu thành 4 cụm (clusters) dựa trên quan sát từ phương pháp Elbow và Silhouette Score trước đó. random_state=42 được dùng để đặt hạt giống ngẫu nhiên, đảm bảo kết quả phân cụm nhất quán giữa các lần chạy.
- Áp dụng K-means lên dữ liệu X_pca đã được giảm chiều. fit_predict thực hiện việc phân cụm và trả về một mảng clusters chứa các nhãn cụm cho từng điểm dữ liệu trong X_pca. Các nhãn cụm này cho biết mỗi điểm dữ liệu thuộc về cụm nào (0, 1, 2 hoặc 3).

```
# PCA và vẽ phân cụm K-means với PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
kmeans = KMeans(n_clusters=4, random_state=42)  # Sử dụng n_clusters=4 theo quan sát từ Elbow/Silhouette
clusters = kmeans.fit_predict(X_pca)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=clusters, palette="viridis", legend='full')
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.title("K-means Clustering with PCA")
plt.show()
```

- Tạo một khung vẽ với kích thước 8x6 inch để chuẩn bị cho việc vẽ biểu đồ.
- Vẽ biểu đồ phân tán (scatter plot) với trục x là thành phần chính thứ nhất (X_pca[:, 0]) và trục y là thành phần chính thứ hai (X_pca[:, 1]).
- hue=clusters cho biết rằng màu sắc của mỗi điểm sẽ dựa trên nhãn cụm của nó.
- Đặt nhãn cho trục x và trục y là "PCA Component 1" và "PCA Component 2"
- Đặt tiêu đề cho biểu đồ là "K-means Clustering with PCA".
- Hiển thị biểu đồ.

3. vẽ biểu đồ radar (radar chart) so sánh cầu thủ

- Chọn hai cầu thủ mà bạn muốn so sánh. Họ cần có cùng vị trí trong đội để so sánh hợp lý.

- Xác định các thuộc tính sẽ dùng để so sánh giữa hai cầu thủ. Mỗi phần tử trong attributes là một bộ ba (tuples), đại diện cho đường dẫn đến cột trong DataFrame. Các thuộc tính này bao gồm "Age", "Matches Played", "non-Penalty Goals", "xG" (Expected Goals), và "SoT" (Shots on Target). Có thể thay đổi các thuộc

```
# Biểu đồ Radar so sánh cầu thủ

# Chọn hai cầu thủ có cùng vị trí để so sánh
p1 = "Elijah Adebayo"  # Tên cầu thủ
p2 = "Simon Adingra"  # Tên cầu thủ

# Danh sách các chỉ số muốn so sánh
attributes = [('','','Age'),('', '', 'Matches Played'),('', 'Performance', 'non-Penalty Goals'),('', 'Expected', 'xG'),('Shooting', 'Standard', 'SoT')]

# Kiểm tra xem cầu thủ và các thuộc tính có trong dữ liệu hay không
if p1 not in df[('', '', 'Name')].values or p2 not in df[('', '', 'Name')].values:
    print(f"One or both players not found in the dataset.")
else:
    # Lấy dữ liệu của hai cầu thủ
    data1 = df[df[('', '', 'Name')] == p1][attributes].values.flatten()
    data2 = df[df[('', '', 'Name')] == p2][attributes].values.flatten()
    print(data1)
    print(data2)
    limits = [40, 50, 10, 1, 40]
    # Kiểm tra nếu có thiếu dữ liệu
    if data1.size == 0 or data2.size == 0:
        print(f"One or both players have missing data for the selected attributes.")
    else:

        scaler = StandardScaler()
        data_combined = np.vstack([data1, data2])  # Kết hợp dữ liệu hai cầu thủ
        data_scaled = scaler.fit_transform(data_combined)

        # Chuẩn bị dữ liệu sau khi chuẩn hóa cho từng cầu thủ
        data1_scaled = np.array([val / lim for val, lim in zip(data1, limits)])
        data2_scaled = np.array([val / lim for val, lim in zip(data2, limits)])
        data1_scaled = np.append(data1_scaled, data1_scaled[0])
        data2_scaled = np.append(data2_scaled, data2_scaled[0])
        # Thiết lập các nhãn và góc cho biểu đồ radar
        labels = [f'{attr[2]}' for attr in attributes]  # Lấy tên chỉ số từ attributes
        num_vars = len(attributes)
        angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
        angles += angles[:1]
```

- Kiểm tra xem cả hai cầu thủ có tồn tại trong cột "Name" của DataFrame không. Nếu không có một trong hai cầu thủ, sẽ xuất thông báo lỗi và ngừng chạy mã.
- Lấy dữ liệu của các thuộc tính cho hai cầu thủ và chuyển thành mảng một chiều (flatten()). Dữ liệu này sẽ dùng cho biểu đồ radar.
- Đặt các giới hạn tối đa cho mỗi chỉ số, để chuẩn hóa dữ liệu vào khoảng 0-1, giúp so sánh dễ dàng hơn.
- Kiểm tra xem có thiếu dữ liệu cho một trong hai cầu thủ không. Nếu có, in ra thông báo lỗi và ngừng thực thi tiếp theo.
- Chuẩn hóa dữ liệu của cả hai cầu thủ thành khoảng giá trị từ 0 đến 1 bằng cách chia giá trị thực tế cho giới hạn tương ứng. Điều này giúp vẽ biểu đồ radar mà không gặp vấn đề về tỷ lệ giữa các chỉ số khác nhau.
- Thêm phần tử đầu tiên của mỗi mảng vào cuối mảng. Điều này giúp kết nối các điểm trên biểu đồ radar, hoàn thành vòng tròn.
- Thiết lập nhãn (labels) và góc cho các thuộc tính trên biểu đồ radar. angles là mảng góc cho mỗi thuộc tính, giúp vẽ biểu đồ radar theo hình tròn.
- Tạo một đồ thị dạng radar (polar plot) với kích thước 6x6 inch.
- Vẽ các đa giác đại diện cho mỗi cầu thủ với màu sắc và độ trong suốt khác nhau, giúp phân biệt các cầu thủ. ax.fill tô màu trong đa giác, còn ax.plot vẽ đường viền.
- Ẩn các nhãn của trục y và chỉ hiển thị nhãn thuộc tính trên trục x, giúp biểu đồ dễ nhìn hơn.
- Đặt tiêu đề biểu đồ và hiển thị chú thích (legend) để phân biệt cầu thủ. Cuối cùng, hiển thị biểu đồ radar.

```
    # Kiểm tra nếu có thiếu dữ liệu
    if data1.size == 0 or data2.size == 0:
        print(f"One or both players have missing data for the selected attributes.")
    else:

        scaler = StandardScaler()
        data_combined = np.vstack([data1, data2])  # Kết hợp dữ liệu hai cầu thủ
        data_scaled = scaler.fit_transform(data_combined)

        # Chuẩn bị dữ liệu sau khi chuẩn hóa cho từng cầu thủ
        data1_scaled = np.array([val / lim for val, lim in zip(data1, limits)])
        data2_scaled = np.array([val / lim for val, lim in zip(data2, limits)])
        data1_scaled = np.append(data1_scaled, data1_scaled[0])
        data2_scaled = np.append(data2_scaled, data2_scaled[0])
        # Thiết lập các nhãn và góc cho biểu đồ radar
        labels = [f'{attr[2]}' for attr in attributes]  # Lấy tên chỉ số từ attributes
        num_vars = len(attributes)
        angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
        angles += angles[:1]

        # Vẽ biểu đồ radar
        fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
        #  # Giới hạn trục y tối đa
        # ax.set_ylim(0, 1.2)  # Thiết lập phạm vi từ 0 đến 1.2 để kiểm soát độ lệch

        ax.fill(angles, data1_scaled, color='blue', alpha=0.25, label=p1)
        ax.fill(angles, data2_scaled, color='red', alpha=0.25, label=p2)
        ax.plot(angles, data1_scaled, color='blue', linewidth=2)
        ax.plot(angles, data2_scaled, color='red', linewidth=2)

        # Thiết lập các nhãn
        ax.set_yticklabels([])
        ax.set_xticks(angles[:-1])
        ax.set_xticklabels(labels)

        # Thêm tiêu đề và chú thích
        plt.title(f"So sánh cầu thủ {p1} và {p2}")
        plt.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))
        plt.show()
```