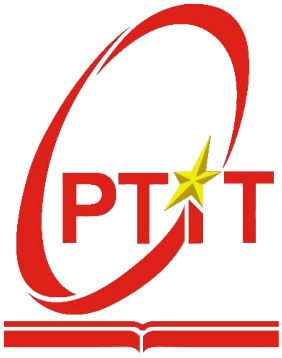
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÀI TẬP LỚN

MÔN : LẬP TRÌNH PYTHON

Họ và tên : **Nguyễn Đình Quang Huy**

Mã sinh viên: B23DCCE046

Lớp: D23CQCE04-B

Hà Nội – 2025

MỤC LỤC

Problem 1

I. Phân tích đề bài

1. Mục tiêu bài toán

Bài toán yêu cầu xây dựng một chương trình thu thập dữ liệu thống kê cầu thủ bóng đá từ Premier League (2024-2025) trên trang web fbref.com. Cụ thể:

- Lấy dữ liệu thống kê của tất cả cầu thủ đã thi đấu hơn 90 phút.
- Lưu kết quả vào file results.csv với những yêu cầu cụ thể về định dạng bảng dữ liệu.

2. Các thông tin cần thu thập

Đề bài yêu cầu lấy một lượng dữ liệu đa dạng và phong phú, bao gồm:

- Thông tin chung: Nation (quốc tịch), Team (đội), Position (vị trí), Age (tuổi).
- Thống kê thi đấu: Matches played (trận), Starts (trận đá chính), Minutes (phút thi đấu).
- Hiệu suất cá nhân: Goals (bàn thắng), Assists (kiến tạo), Yellow Cards, Red Cards.
- Thống kê kỳ vọng (Expected): Expected Goals (xG), Expected Assists (xAG).
- Khả năng tạo đột phá: Progressive Carries, Passes, Receptions (PrgC, PrgP, PrgR).
- Hiệu suất trung bình trên 90 phút: Goals/90, Assists/90, xG/90, xAG/90.
- Các mảng khác: Kỹ năng sút, chuyền bóng, phòng ngự, cầm bóng, thi đấu hỗn hợp (Misc).

→ Tổng cộng 77 chỉ số thống kê.

3. Ràng buộc kỹ thuật

- Nguồn dữ liệu: https://fbref.com/en/
- Điều kiện lọc: chỉ cầu thủ > 90 phút.
- Xử lý dữ liệu thiếu: Ghi là "N/a".
- Sắp xếp: Theo First Name (tên đầu tiên).
- Đầu ra: File CSV duy nhất results.csv.

4. Thách thức kỹ thuật

- Web scraping nhiều bảng dữ liệu khác nhau từ fbref.
- Chuẩn hóa dữ liệu từ nhiều bảng, nhiều định dạng khác nhau.
- Merge dữ liệu cầu thủ theo tên + đội bóng (để tránh trùng lặp).
- Đảm bảo dữ liệu đầy đủ, không lỗi cú pháp khi lưu CSV.

II. Các công cụ và thư viện sử dụng

Để giải quyết bài toán này, ta cần các công cụ chuyên dụng cho web scraping, xử lý dữ liệu bảng, và lưu trữ dữ liệu. Cụ thể:

Công cụ	Vai trò	Tại sao dùng ?
Selenium	Truy cập và điều khiển trình duyệt, tải trang web động(Javascript rendering)	Fbref sử dụng Javascript để render table → BeautifulSoup / Requests không đủ
BeautifulSoup	Parse HTML lấy nội dung table (DOM parsing)	Trích xuất dữ liệu từ mã HTML
Pandas	Xử lý bảng dữ liệu, merge, fill NA, sort, xuất CSV	Quản lý bảng cầu thủ/phép toán DataFrame
Webdriver-manager	Tự động cài đặt và quản lý ChromeDriver	Đỡ phải tự cài đặt driver thủ công
Selenium Expected Conditions	Chờ trang load table trước khi scrape	Giảm lỗi "element chưa xuất hiện"

III. Code và giải thích chi tiết

1. Đầu tiên ,em import thư viện và thiết lập driver

```python
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"

```

2. Sau đó ,em xây dựng hàm lấy dữ liệu từ trang web

```python
# Hàm lấy dữ liệu từ web
def GetDataFromWeb(url, Xpath_player, Data_name):
    service = Service(ChromeDriverManager().install())
    driver1 = webdriver.Chrome(service=service)
    driver1.get(url)
    player_list = []
    try:
        WebDriverWait(driver1, 10).until(EC.presence_of_element_located((By.XPATH, Xpath_player)))
        table_element = driver1.find_element(By.XPATH, Xpath_player)
        html_table = table_element.get_attribute('outerHTML')
        soup = bs(html_table, 'html.parser')
        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()
                        data.append(a[1] if len(a) == 2 else 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:
        driver1.quit()
        print("Finish " + Data_name)
    return player_list

```

Hàm GetDataFromWeb được viết để:

- Mở trang web chứa bảng thống kê.
- Tìm đúng bảng dữ liệu bằng Xpath.
- Lấy mã HTML của bảng, sau đó parse bằng BeautifulSoup.
- Duyệt từng dòng trong bảng để lấy dữ liệu cầu thủ.
- Làm sạch dữ liệu (chuyển đổi số, xử lý giá trị trống).
- Cuối cùng đóng trình duyệt và trả về danh sách dữ liệu.

3. Thu thập từng nhóm dữ liệu cầu thủ

- Em đã lần lượt thu thập 7 bảng thống kê chính trên fbref:
  + Standard stats (Thống kê cơ bản)
  + Keepers stats (Thủ môn)
  + Shooting stats (Sút bóng)
  + Passing stats (Chuyền bóng)
  + GCA stats (Tạo cơ hội ghi bàn)
  + Defense stats (Phòng ngự)
  + Possession stats (Kiểm soát bóng)
  + Misc stats (Thống kê hỗn hợp)
- Với mỗi bảng, em truyền vào url, Xpath, Data_name và thu thập danh sách dữ liệu tương ứng.

```python
# stats_keeper
url = "https://fbref.com/en/comps/9/2024-2025/keepers/2024-2025-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]
    GA90, Save_perc, CS_perc, PKSave_perc = p[11], p[14], p[16], p[20]
    keeper_list.append([Name,Team,GA90, Save_perc, CS_perc, PKSave_perc])
df_keepers = pd.DataFrame(keeper_list, columns=['Name','Team','GA90', 'Save%', 'CS%', 'PK Save%'])

```

4. Xử lý và chuẩn hóa dữ liệu từ từng bảng

- Ở mỗi bảng, em lọc ra đúng các chỉ số cần thiết, bỏ những cột không cần dùng.
- Với bảng Standard, em lọc cầu thủ có số phút thi đấu > 90 đúng yêu cầu đề bài.

```python
# stats_standard
url = "https://fbref.com/en/comps/9/2024-2025/stats/2024-2025-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]
        Gls, Ast, ycard, rcard = p[9:13]
        xG, xAG = p[18], p[20]
        PrgC, PrgP, PrgR = p[22:25]
        Gls90, Ast90, xG90, xAG90 = p[25], p[26], p[30], p[31]
        if min > 90:
            player_list.append([Name, Nation, Team, Position, Age, mp, starts, min, Gls, Ast, ycard, rcard,
    except IndexError:
        break

cols_player = ['Name','Nation','Team','Position','Age','Matches Played','Starts','Min','Goals','Assists','Y
df_player = pd.DataFrame(player_list, columns=cols_player)

```

5. Gộp tất cả bảng dữ liệu vào một bảng duy nhất

Sau khi thu thập được dữ liệu từ nhiều bảng thống kê khác nhau (Standard, Keepers, Shooting, Passing, GCA, Defense, Possession, Misc), em tiến hành **kết hợp (merge)** các bảng này lại thành một bảng tổng hợp duy nhất.

Quy trình thực hiện như sau:

- Em xác định khóa chung (key) để kết hợp dữ liệu là hai cột: Name (tên cầu thủ) và Team (đội bóng).

→ Điều này giúp đảm bảo dữ liệu của một cầu thủ trong một đội được ghép đúng nhau.

- Em lần lượt merge từng DataFrame với DataFrame chính (df_player) theo kiểu left join.

→ Kiểu left join đảm bảo rằng tất cả cầu thủ trong bảng chính (Standard stats — cầu thủ >90 phút) đều giữ nguyên, dù họ có hay không có dữ liệu ở bảng phụ (ví dụ: có cầu thủ không có chỉ số thủ môn, Shooting,…)

- Cụ thể, tôi tạo danh sách các DataFrame phụ và dùng vòng lặp để merge từng cái một:

```
# Merge tất cả lại
dataframes = [df_keepers,df_shooting,df_Passing, df_GCA,df_defense,df_possession, df_misc]
df_merged = df_player
for df in dataframes:
    df_merged = pd.merge(df_merged,df, on = ['Name','Team'], how = 'left')

```

- Sau khi merge xong, DataFrame df_merged sẽ chứa toàn bộ chỉ số của từng cầu thủ, trải dài trên 77 cột, bao phủ tất cả nhóm thống kê mà đề bài yêu cầu.

6. Sắp xếp dữ liệu và lưu file kết quả

- Sau khi có bảng tổng hợp dữ liệu df_merged, em thực hiện bước cuối cùng là sắp xếp dữ liệu và lưu file kết quả.
- Quy trình thực hiện như sau:

+ Tách tên đầu tiên (First Name): Đề bài yêu cầu sắp xếp cầu thủ theo tên đầu tiên (First Name), nên em sử dụng hàm split() để lấy từ đầu tiên trong cột Name.

+ Sắp xếp cầu thủ: Em sắp xếp DataFrame theo First Name (tăng dần) và Age (giảm dần) để đảm bảo đúng yêu cầu bài toán.

+ Xóa cột phụ (First Name): Sau khi sắp xếp xong, em xóa cột phụ này để bảng kết quả gọn gàng.

+ Lưu file CSV: Em lưu DataFrame kết quả vào file results.csv. Để đảm bảo yêu cầu "nếu dữ liệu thiếu thì ghi N/a", em dùng tham số na_rep='N/a'.

- Cuối cùng, chương trình in ra thông báo: print('Du lieu da duoc luu vao file results.csv')

```
# Sắp xếp và lưu
df_merged['First Name'] = df_merged['Name'].apply(lambda x: x.split()[0])
df_sorted = df_merged.sort_values(by=['First Name', 'Age'], ascending=[True, False])
df_sorted = df_sorted.drop(columns=['First Name'])
df_sorted.to_csv('results.csv', index=False, na_rep='N/a')

print('Du lieu da duoc luu vao file results.csv')
```

→ **Kết quả**: File results.csv chứa dữ liệu đầy đủ của tất cả cầu thủ đã thi đấu >90 phút tại Premier League mùa 2024–2025, đúng định dạng yêu cầu đề bài.

IV. Kết quả

- File đầu ra: results.csv
- Bảng dữ liệu thống kê của tất cả cầu thủ đá >90 phút Premier League 2024-2025.
- Số lượng chỉ số: 77 chỉ số, bao quát tất cả mảng: cá nhân, phòng ngự, tấn công, chuyền bóng, cầm bóng, thủ môn, tranh chấp,...

- Tính đúng yêu cầu đề bài:
  + Lọc đúng cầu thủ >90 phút.
  + Xử lý thiếu NA.
  + Merge đúng tên + đội bóng.
  + Sắp xếp đúng thứ tự.
  + Lưu đúng file CSV.
- Ứng dụng thực tế: Bộ dữ liệu này là nền tảng cho các bài toán phân tích hiệu suất, so sánh cầu thủ, xếp hạng đội bóng,tính giá trị chuyển nhượng sau (Bài 2, 3, 4 tiếp theo).

Problem 2

I. Phân tích đề bài

- Bài toán yêu cầu viết chương trình Python để phân tích thống kê dữ liệu cầu thủ đã thu thập từ Bài 1 (file results.csv). Cụ thể:
  + Xác định Top 3 cầu thủ cao nhất/thấp nhất của từng chỉ số.
  + Tính Median, Mean, Std cho từng chỉ số — cho toàn bộ và cho từng đội bóng.
  + Vẽ biểu đồ histogram cho từng chỉ số — cho toàn bộ và từng đội.
  + Xác định đội mạnh nhất cho từng chỉ số và đội toàn diện nhất (nhiều chỉ số đứng đầu nhất).
- Các kết quả cần lưu:
  + File top_3.txt (top 3 cầu thủ)
  + File results2.csv (Median/Mean/Std)
  + Các ảnh biểu đồ histogram
  + File team_best_stats.txt (đội mạnh nhất mỗi chỉ số)

II. Các công cụ và thư viện sử dụng

Để giải quyết bài toán này, ta cần các công cụ chuyên dụng cho xử lý dữ liệu bảng, phân tích thống kê, vẽ biểu đồ và lưu trữ kết quả. Cụ thể:

Công cụ	Vai trò	Tại sao dùng ?
Pandas	Xử lý bảng dữ liệu, tính toán thống kê (mean, median, std) , lọc và nhóm dữ liệu	Hỗ trợ thao tác DataFrame nhanh và linh hoạt
NumPy	Xử lý giá trị NaN, chuyển đổi kiểu dữ liệu số	Giúp xử lý dữ liệu thiếu và chuẩn hóa dữ liệu đầu vào
Matplotlib	Vẽ biểu đồ histogram trực quan cho các chỉ số	Giúp trực quan hóa phân bố thống kê

OS	Tạo thư mục lưu ảnh biểu đồ	Tự động hóa việc lưu file hình ảnh vào đúng thư mục
Python built-in (open, with)	Đọc/ghi file .txt, .csv	Lưu trữ kết quả phân tích dưới dạng file để dễ nộp bài

III. Code và giải thích chi tiết

1. Đọc dữ liệu và chuẩn hóa

- Em sử dụng Pandas để đọc file kết quả results.csv từ Bài 1.
- Sau đó em chuẩn hóa dữ liệu như sau:

+ Thay giá trị "N/a" bằng NaN để có thể thực hiện phép toán (vì "N/a" là dạng chuỗi, không tính toán được).

+ Chuyển tất cả các cột dữ liệu có thể thành kiểu float để đảm bảo có thể tính toán Mean, Median, Std.

→ Mục đích: Đảm bảo dữ liệu sẵn sàng cho các phép toán thống kê.

```
6   # Đọc dữ liệu
7   df = pd.read_csv('results.csv')
8
9   # Chuyển 'N/a' thành NaN để tính toán không bị lỗi
10  df.replace('N/a', np.nan, inplace=True)
11
12  # Chuyển tất cả cột số thành float (nếu có thể)
13  for col in df.columns:
14      try:
15          df[col] = df[col].astype(float)
16      except:
17          continue
18
```

2. Xác định các cột số (numeric)

- Loại bỏ cột không phải số (Name, Nation, Team, Position).
- Danh sách cột số sẽ được dùng cho các bước tiếp theo.

```
18
19  # Lọc danh sách cột số (numeric)
20  exclude_cols = ['Name', 'Nation', 'Team', 'Position']
21  numeric_cols = [col for col in df.columns if col not in exclude_cols and pd.api.types.is_numeric_dtype(df[col])]
22
23  # Bước 1 - Top 3 cao nhất / thấp nhất
```

3. Tìm Top 3 cao nhất/thấp nhất từng chỉ số

- Với mỗi chỉ số (trong numeric_cols), em:

+ Bỏ qua giá trị NaN (cầu thủ không có chỉ số đó).

+ Sắp xếp giảm dần để lấy Top 3 cao nhất.

+ Sắp xếp tăng dần để lấy Top 3 thấp nhất.

- Em ghi kết quả ra file top_3.txt dưới định dạng dễ đọc.

→ Mục đích: Giúp so sánh cầu thủ nổi bật nhất và kém nhất ở từng chỉ số.

```
# Top 3 cao nhất / thấp nhất

with open('top_3.txt', 'w', encoding='utf-8') as f:
    for col in numeric_cols:
        f.write(f'\n==== {col} ====\n')
        # Bỏ NaN khi sort
        top3 = df[['Name','Team',col]].dropna(subset=[col]).sort_values(by=col, ascending=False).head(3)
        bottom3 = df[['Name','Team',col]].dropna(subset=[col]).sort_values(by=col, ascending=True).head(3)

        f.write('\nTop 3 highest:\n')
        f.write(top3.to_string(index=False))
        f.write('\n\nTop 3 lowest:\n')
        f.write(bottom3.to_string(index=False))
        f.write('\n\n')

print('Đã lưu file top_3.txt')

```

4. Tính Median, Mean, Std cho từng chỉ số

- Em thực hiện hai nhóm tính toán:

+ Cho toàn bộ cầu thủ (Teams = all): Tính Median, Mean, Std cho từng chỉ số trên toàn bộ cầu thủ.

+ Cho từng đội bóng: Tính Median, Mean, Std cho từng chỉ số nhưng lọc riêng từng đội bóng.

→ Mục đích: Giúp so sánh trung vị, trung bình và độ lệch chuẩn của từng chỉ số giữa toàn giải và từng đội.

- Cách làm:

+ Lặp qua từng đội (dùng .unique() trên cột Team).

+ Lưu kết quả vào danh sách summary_rows.

+ Sau cùng chuyển thành DataFrame và xuất file results2.csv.

```

# Tính Median, Mean, Std (all + từng team)

summary_rows = []

# Cho toàn bộ
row_all = ['all']
for col in numeric_cols:
    row_all.extend([
        df[col].median(skipna=True),
        df[col].mean(skipna=True),
        df[col].std(skipna=True)
    ])
summary_rows.append(row_all)

# Cho từng đội
teams = df['Team'].dropna().unique()

for team in teams:
    df_team = df[df['Team'] == team]
    row = [team]
    for col in numeric_cols:
```

```
            row.extend([
                df_team[col].median(skipna=True),
                df_team[col].mean(skipna=True),
                df_team[col].std(skipna=True)
            ])
        summary_rows.append(row)

    # Lưu file results2.csv
    columns = ['Team']
    for col in numeric_cols:
        columns.extend([f'Median_{col}', f'Mean_{col}', f'Std_{col}'])

    df_summary = pd.DataFrame(summary_rows, columns=columns)
    df_summary.to_csv('results2.csv', index=False)

    print('Đã lưu file results2.csv')

```

5. Vẽ biểu đồ histogram cho 6 chỉ số tiêu biểu

- Em chọn 6 chỉ số đại diện: Tấn công (Goals, Assists, xG) + Phòng ngự (Tkl, Blocks, Int)
- Với mỗi chỉ số, em vẽ 2 loại biểu đồ:
  + Histogram cho toàn bộ cầu thủ.
  + Histogram cho từng đội bóng.
- Ảnh biểu đồ lưu vào thư mục histograms/ (sẽ tự động tạo nếu chưa có).

```
# Chỉ số tấn công và phòng thủ được chọn
attack_cols = ['Goals', 'Assists', 'xG']
defense_cols = ['Tkl', 'Blocks', 'Int']
selected_cols = attack_cols + defense_cols

os.makedirs('histograms', exist_ok=True)

for col in selected_cols:
    if col not in df.columns:
        print(f"Cột {col} không tồn tại trong dữ liệu – bỏ qua")
        continue

    plt.figure(figsize=(8,5))
    plt.hist(df[col].dropna(), bins=20, color='blue', alpha=0.7)
    plt.title(f'Distribution of {col} (All players)')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.tight_layout()
    plt.savefig(f'histograms/{col}_all.png')
    plt.close()

    for team in teams:
        df_team = df[df['Team'] == team]
        plt.figure(figsize=(8,5))
        plt.hist(df_team[col].dropna(), bins=20, color='green', alpha=0.7)
        plt.title(f'Distribution of {col} ({team})')
        plt.xlabel(col)
        plt.ylabel('Frequency')
        plt.tight_layout()
        filename = f'histograms/{col}_{team.replace("/", "-")}.png'
        plt.savefig(filename)
        plt.close()

print('Đã vẽ xong histogram cho 6 chỉ số (thư mục histograms)')

```

6. Xác định đội mạnh nhất từng chỉ số

- Em dùng Pandas groupby để nhóm dữ liệu theo Team.
- Với từng chỉ số, em tính giá trị trung bình (mean) của từng đội.
- Sau đó, em dùng .idxmax() để xác định đội có trung bình cao nhất và .max() để lấy giá trị cụ thể.
- Kết quả lưu vào dictionary team_best
- Cuối cùng em ghi toàn bộ kết quả vào file team_best_stats.txt dưới dạng dễ đọc.

```
115  # Tìm đội có giá trị cao nhất cho từng chỉ số
116
117  team_best = {}
118
119  for col in numeric_cols:
120      team_avg = df.groupby('Team')[col].mean()
121      best_team = team_avg.idxmax()
122      best_value = team_avg.max()
123      team_best[col] = (best_team, best_value)
124
125  # Lưu bảng tổng hợp đội mạnh nhất theo từng chỉ số
126  with open('team_best_stats.txt', 'w', encoding='utf-8') as f:
127      for stat, (team, value) in team_best.items():
128          f.write(f'{stat}: Best team = {team} | Avg = {value}\n')
129
130  print('Đã lưu file team_best_stats.txt (thống kê đội mạnh nhất mỗi chỉ số)')
131
```

7. Xác định đội toàn diện nhất

- Em duyệt lại toàn bộ danh sách chỉ số → đội mạnh nhất (dictionary team_best từ bước 6).
- Với mỗi đội xuất hiện, em tăng điểm (cộng 1) vào bảng điểm team_score.
- Sau khi đếm xong, em tìm đội có số điểm cao nhất → đây chính là đội toàn diện nhất.
- Cuối cùng, em ghi kết quả vào cuối file top_3.txt, bao gồm:
  + Tên đội thắng cuộc
  + Tổng số điểm (số lần đứng đầu)
  + Bảng điểm chi tiết của từng đội

```python
# Xác định đội toàn diện nhất

# Tính điểm mỗi đội (số lần đứng đầu ở các chỉ số)
team_score = {}

for col in numeric_cols:
    team_avg = df.groupby('Team')[col].mean()
    best_team = team_avg.idxmax()
    if pd.notna(best_team):
        team_score[best_team] = team_score.get(best_team, 0) + 1

# Đội có điểm cao nhất
best_overall_team = max(team_score, key=team_score.get)
best_overall_score = team_score[best_overall_team]

# Ghi kết quả vào cuối file top_3.txt
with open('top_3.txt', 'a', encoding='utf-8') as f:
    f.write('\n' + '='*40 + '\n')
    f.write(f'\nĐỘI TOÀN DIỆN NHẤT (THE BEST OVERALL TEAM)\n')
    f.write(f'\n{best_overall_team} với tổng điểm {best_overall_score} (số lần đứng đầu chỉ số)\n')
    f.write('\nChi tiết điểm từng đội:\n')
    for team, score in sorted(team_score.items(), key=lambda x: x[1], reverse=True):
        f.write(f'{team}: {score} điểm\n')

print(f'Đội toàn diện nhất là {best_overall_team} (đã ghi vào top_3.txt)')

```

IV. Kết quả

Sau khi thực hiện đầy đủ các bước xử lý và phân tích dữ liệu, chương trình đã cho ra các kết quả cụ thể như sau:

1. File top_3.txt — Top 3 cầu thủ cao nhất/thấp nhất từng chỉ số

- File này liệt kê Top 3 cầu thủ có chỉ số cao nhất và Top 3 cầu thủ có chỉ số thấp nhất cho từng chỉ số đã thu thập ở Bài 1 (tổng cộng ~77 chỉ số).
- Kết quả bao gồm tên cầu thủ, đội bóng và giá trị chỉ số.
- Ví dụ

==== Matches Played ====

Top 3 highest:

Name	Team	Matches Played
Youri Tielemans	Aston Villa	34.0
Virgil van Dijk	Liverpool	34.0
Tyrick Mitchell	Crystal Palace	34.0

Top 3 lowest:

Name	Team	Matches Played
Odsonne Édouard	Crystal Palace	2.0
Ayden Heaven	Manchester Utd	2.0
Neto	Bournemouth	2.0

- Ý nghĩa: Giúp xác định nhanh cầu thủ nổi bật nhất và kém nhất ở từng khía cạnh.

2. File results2.csv — Median, Mean, Std cho toàn bộ và từng đội

- Bảng tổng hợp giá trị trung vị (Median), trung bình (Mean), độ lệch chuẩn (Std) cho từng chỉ số:
  + Cho toàn bộ cầu thủ (Team = all)
  + Cho từng đội bóng (20 đội)
- Bảng có dạng**:**

Team	Median_Goals	Mean_Goals	Std_Goals	Median_Assists	Mean_Assists	…
all	…	…	…	…	…	…
Arsenal	…	…	…	…	…	…
Liverpool	…	…	…	…	…	…
…	…	…	…	…	…	…

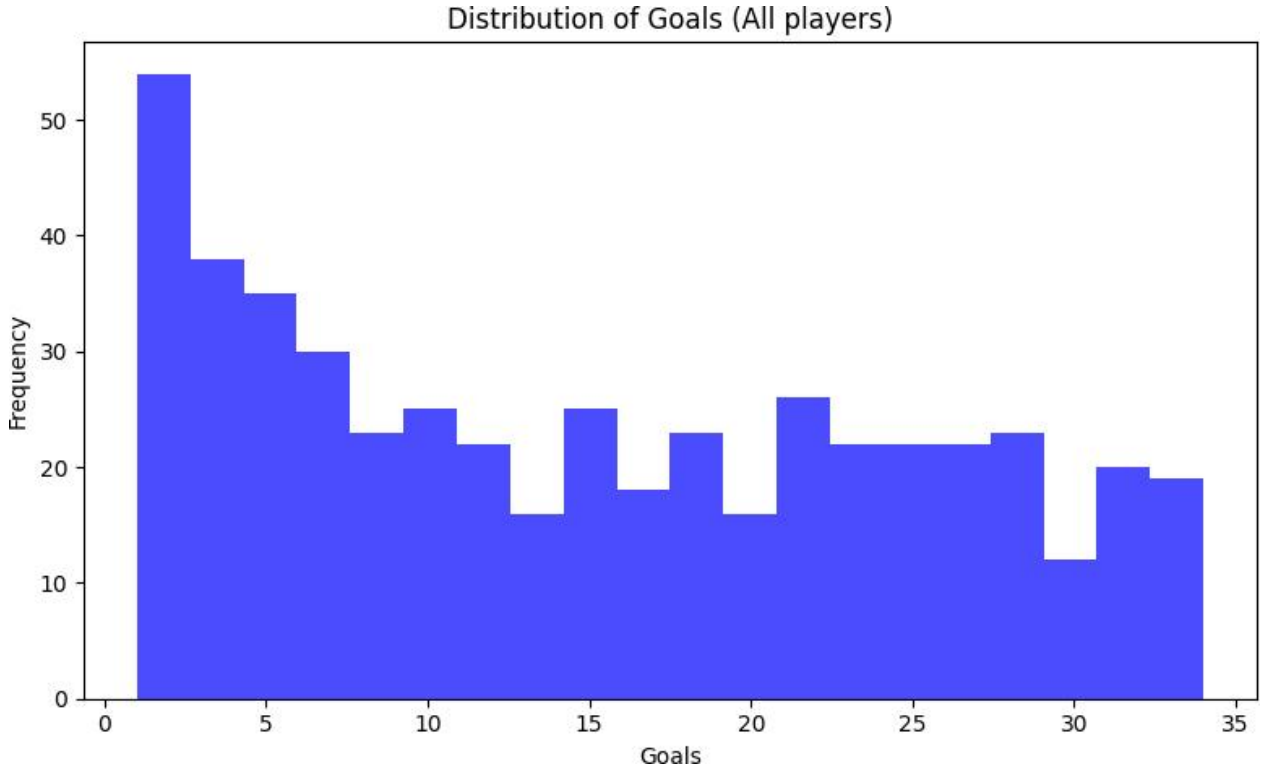
- Ý nghĩa: Giúp so sánh mức độ trung bình và ổn định từng chỉ số giữa các đội và toàn giải.

3. Thư mục histograms/ — Biểu đồ phân phối chỉ số

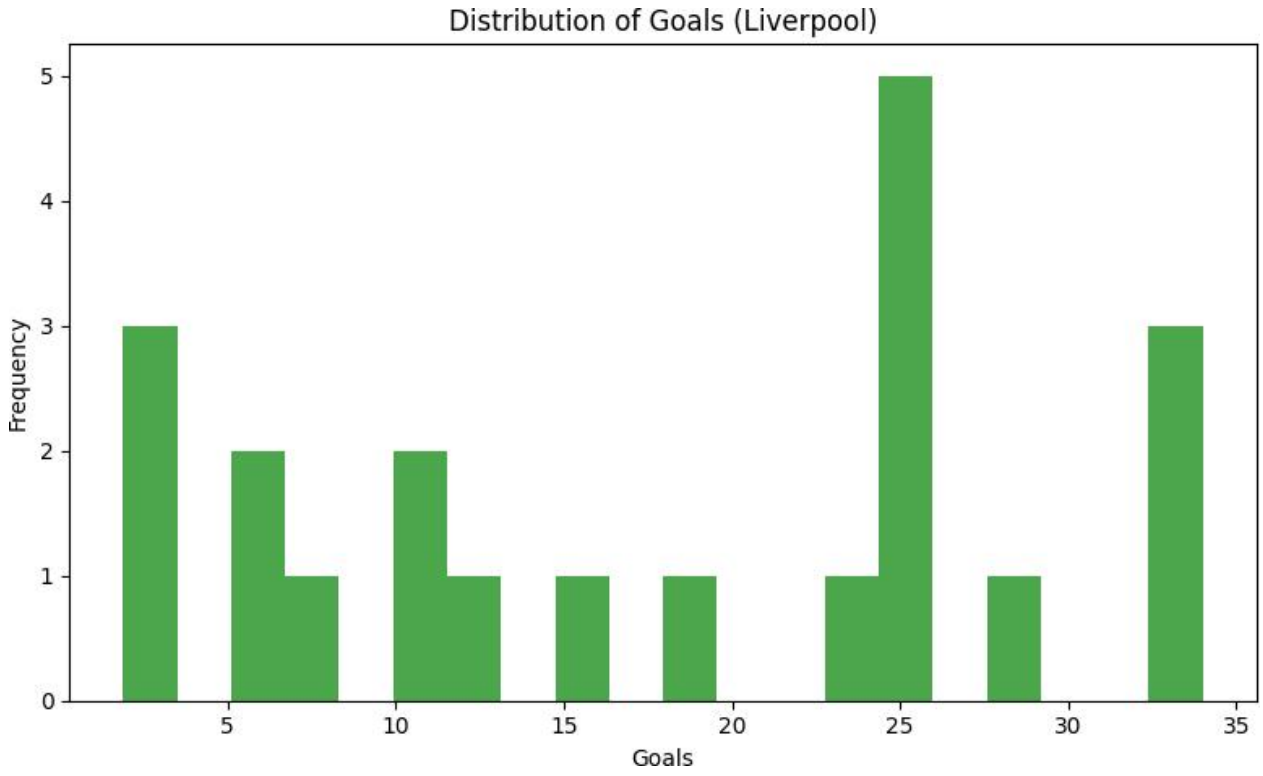
- Thư mục này chứa 36 biểu đồ histogram (6 chỉ số × (1 biểu đồ toàn giải + 20 biểu đồ đội bóng)).
- Các biểu đồ thể hiện phân bố tần suất từng chỉ số của cầu thủ.

Ví dụ ảnh:

+ Goals_all.png: Phân bố số bàn thắng của toàn bộ cầu thủ.



+ Goals_Liverpool.png: Phân bố số bàn thắng của cầu thủ Liverpool.



- Ý nghĩa: Giúp trực quan hóa sự phân bố của từng chỉ số — phát hiện đội nào vượt trội hoặc đồng đều hơn.

4. File team_best_stats.txt — Đội mạnh nhất từng chỉ số

- File này liệt kê đội có giá trị trung bình cao nhất ở từng chỉ số.
- Ví dụ kết quả:

Goals: Best team = Liverpool | Avg = 17.747619047619047
Assists: Best team = Liverpool | Avg = 3.761904761904762
xG: Best team = Liverpool | Avg = 3.6285714285714286

Ý nghĩa: Giúp xác định đội nào dẫn đầu từng khía cạnh thi đấu (tấn công, phòng ngự, kiến tạo…).

5. Đội toàn diện nhất

- Kết quả được ghi cuối file top_3.txt.
- Liverpool là đội toàn diện nhất với 27 lần đứng đầu trên tổng số 77 chỉ số.

```
1110  ======================================
1111
1112  ĐỘI TOÀN DIỆN NHẤT (THE BEST OVERALL TEAM)
1113
1114  Liverpool với tổng điểm 27 (số lần đứng đầu chỉ số)
1115
1116  Chi tiết điểm từng đội:
1117  Liverpool: 27 điểm
1118  Manchester City: 12 điểm
1119  Bournemouth: 6 điểm
1120  Crystal Palace: 6 điểm
1121  Brentford: 4 điểm
1122  Fulham: 3 điểm
1123  Leicester City: 2 điểm
1124  Everton: 2 điểm
1125  Nott'ham Forest: 2 điểm
1126  Arsenal: 2 điểm
1127  Tottenham: 2 điểm
1128  Newcastle Utd: 2 điểm
1129  Aston Villa: 1 điểm
1130  Chelsea: 1 điểm
1131  Manchester Utd: 1 điểm
1132
```

→ Ý nghĩa: Cho thấy Liverpool là đội bóng ổn định, đồng đều và mạnh nhất xét trên toàn bộ các chỉ số của mùa giải 2024–2025.

Problem 3

I. Phân tích đề bài

Yêu cầu bài toán:

- Dựa theo file PDF assignment_1_E23.pdf, bài 3 yêu cầu thực hiện phân cụm dữ liệu người chơi (players) bằng thuật toán K-means clustering. Sau đó:
  + Tìm số lượng cụm tối ưu (k) bằng phương pháp Elbow Method.
  + Ánh xạ dữ liệu sau phân cụm thành không gian 2 chiều bằng PCA (Principal Component Analysis) để trực quan hóa kết quả.
  + Lưu kết quả phân cụm ra file CSV.
- Dữ liệu đầu vào:file results.csv.
- Dữ liệu đầu ra:
  + Biểu đồ Elbow Method (WCSS theo k).
  + Biểu đồ scatter plot 2D các cụm sau khi giảm chiều với PCA.
  + File CSV chứa dữ liệu gốc kèm theo nhãn cụm.

II. Các công cụ và thư viện sử dụng

Trong bài toán này, các thư viện và công cụ cụ thể được sử dụng nhằm phục vụ từng bước nhỏ trong quy trình phân cụm và trực quan hóa:

Công cụ/ Thư viện	Vai trò	Tại sao sử dụng ?

Pandas	Đọc, xử lý và lưu trữ dữ liệu dạng bảng (CSV).	Hỗ trợ các thao tác với DataFrame như thêm cột cluster.
NumPy	Hỗ trợ tính toán số học, đặc biệt liên quan đến mảng dữ liệu (arrays).	Chủ yếu phục vụ nội bộ trong pandas/sklearn.
matplotlib.pyplot	Vẽ biểu đồ: Elbow Method và Scatter Plot (PCA visualization).	Giúp trực quan hóa trực tiếp sự phân cụm và chọn số lượng cụm.
sklearn.preprocessing. StandardScaler	Chuẩn hóa dữ liệu trước khi KMeans clustering.	Đảm bảo mỗi feature có trung bình = 0, độ lệch chuẩn = 1, giúp phân cụm chính xác hơn.
sklearn.cluster.KMeans	Thực hiện thuật toán phân cụm K-means.	Là công cụ trọng tâm của bài.
sklearn.decomposition.PCA	Giảm chiều dữ liệu từ N chiều xuống 2 chiều để vẽ scatter plot.	Trực quan hóa dữ liệu phức tạp một cách dễ hiểu.

- Nhận xét :

+ Việc chuẩn hóa dữ liệu (standardization) là cực kỳ cần thiết trước khi áp dụng K-means và PCA, vì:

+ Nếu không chuẩn hóa, những biến có giá trị lớn (ví dụ: số phút thi đấu) sẽ chi phối toàn bộ thuật toán, khiến phân cụm bị méo mó.

+ Matplotlib hỗ trợ trực quan hóa cực kỳ hiệu quả cho phân tích dữ liệu, đặc biệt trong đánh giá điểm gãy của Elbow Method.

III. Phân tích code chi tiết

1. Đọc ,tiền xử lý dữ liệu và chuẩn hóa dữ liệu

- Mục đích:

+ Đọc dữ liệu từ file results.csv.

+ Lọc ra chỉ những cột số (numeric columns) để phù hợp với KMeans (vì text không thể phân cụm trực tiếp).

+ Thay thế các giá trị thiếu (NaN) bằng 0.

+ Đưa tất cả các đặc trưng về cùng thang đo.

```python
# 1. Đọc dữ liệu
data = pd.read_csv('results.csv')

# 2. Tiền xử lý dữ liệu
# Loại bỏ cột text nếu có (Nation, Team, Position)
data_numeric = data.select_dtypes(include=[np.number])

# Xử lý giá trị thiếu (nếu có)
data_numeric = data_numeric.fillna(0)

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data_numeric)

```

- Nhận xét:

+ Đây là cách làm đơn giản nhưng đôi khi gán 0 cho giá trị thiếu có thể làm sai lệch dữ liệu thực tế.

+ Một hướng tốt hơn có thể là gán giá trị trung bình hoặc dùng kỹ thuật Imputation.

+ Dữ liệu sau chuẩn hóa có trung bình = 0 và độ lệch chuẩn = 1.

+ Giúp tránh hiện tượng "thống trị" của các feature lớn.

2. Xác định số lượng cụm tối ưu (k) với Elbow Method

```python
# *** Vẽ biểu đồ Elbow để chọn số cụm tối ưu ***
wcss = []
k_range = range(1, 11)
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(8,6))
plt.plot(k_range, wcss, 'o-', marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.grid(True)
plt.show()

```

- WCSS (Within Cluster Sum of Squares): Tổng bình phương khoảng cách từ mỗi điểm dữ liệu tới tâm cụm của nó.

- Elbow Method:

+ Vẽ đồ thị WCSS theo số lượng cụm k.

+ Quan sát "góc gãy" (elbow) để xác định k tối ưu.

+ Dựa trên hình Elbow (Figure_1.png), k = 4 là lựa chọn hợp lý vì sau k=4, WCSS giảm chậm.

3. Phân cụm dữ liệu bằng KMeans

```
37
38  # 3. Áp dụng K-means với K=4
39  kmeans = KMeans(n_clusters=4, random_state=42)
40  kmeans.fit(scaled_data)
41  labels = kmeans.labels_
42
43  # Thêm nhãn cụm vào dữ liệu gốc
44  data['Cluster'] = labels
45
```

- Mục tiêu: Chia dữ liệu thành 4 cụm dựa trên sự tương đồng.
- Cách hoạt động:
  + Kmeans chọn 4 tâm cụm ban đầu ngẫu nhiên (với random_state đảm bảo tái lập).
  + Lặp lại: gán nhãn cụm + cập nhật tâm mới cho tới khi hội tụ.
- Kết quả: Mỗi người chơi được gán vào một trong bốn nhóm từ Cluster 0 → Cluster 3.

4. Giảm chiều dữ liệu bằng PCA và trực quan hóa

```
46  # Giảm chiều dữ liệu bằng PCA (2D)
47  pca = PCA(n_components=2)
48  pca_data = pca.fit_transform(scaled_data)
49
50  # Thêm PCA vào dataframe để tiện vẽ
51  pca_df = pd.DataFrame(data=pca_data, columns=['PCA1', 'PCA2'])
52  pca_df['Cluster'] = labels
53
54  # Vẽ biểu đồ scatter 2D
55  plt.figure(figsize=(12,6))
56  colors = ['red', 'blue', 'green', 'orange']
57  for cluster in range(4):
58      cluster_points = pca_df[pca_df['Cluster'] == cluster]
59      plt.scatter(cluster_points['PCA1'], cluster_points['PCA2'],
60                  c=colors[cluster], label=f'Cluster {cluster}', alpha=0.6)
61
62  plt.title('2D PCA clusters visualization of Players')
63  plt.xlabel('PCA1')
64  plt.ylabel('PCA2')
65  plt.legend()
66  plt.grid(True)
67  plt.show()
68
```

- Scatter plot (Figure_2.png) thể hiện rõ:
  + Các nhóm cầu thủ có phân tách rõ ràng theo các cụm.
  + Các cụm phân bố hợp lý, không quá chồng chéo.

5. Lưu kết quả

```
69  # (Tùy chọn) Xuất file kết quả
70  data.to_csv('results_with_clusters.csv', index=False)
71
72  print("Đã phân cụm xong, số lượng cầu thủ mỗi cụm:")
73  print(data['Cluster'].value_counts())
74
```

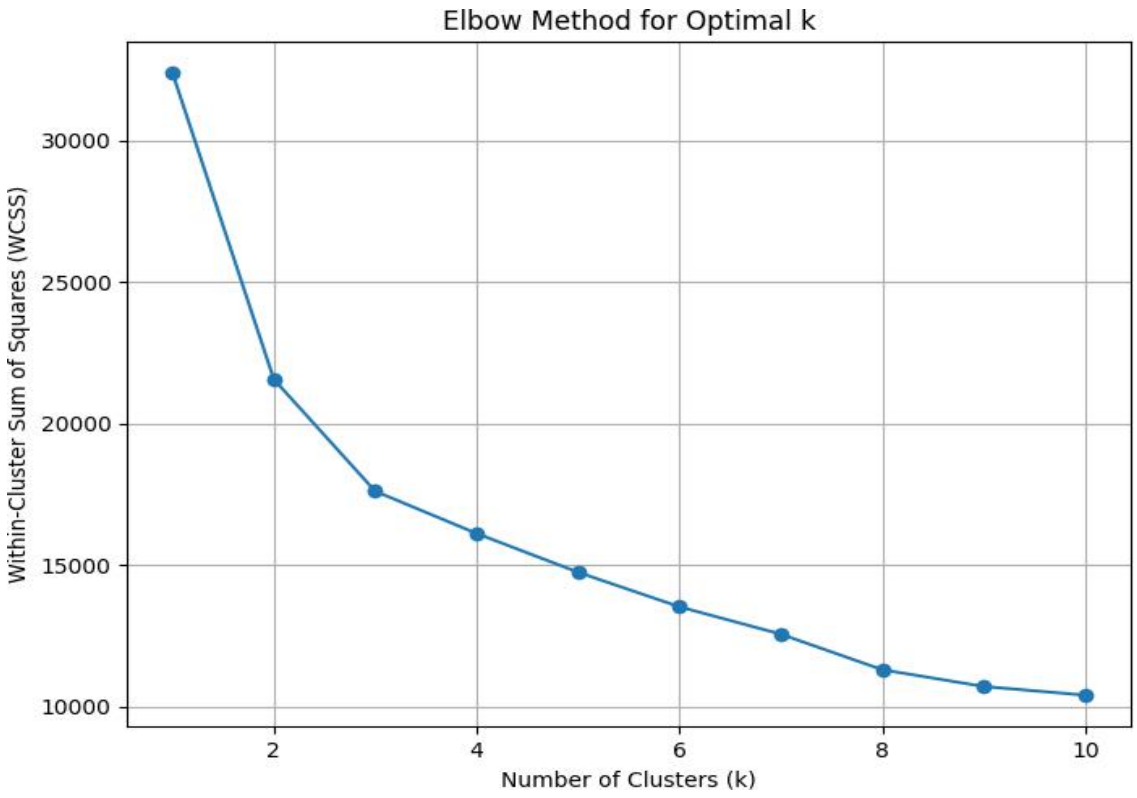
- Lưu lại toàn bộ bảng dữ liệu + cột Cluster vào file results_with_clusters.csv.
- In thêm thống kê số lượng cầu thủ trong từng cụm:

IV. Kết quả

1. Xác định số lượng cụm (k) tối ưu bằng Elbow Method

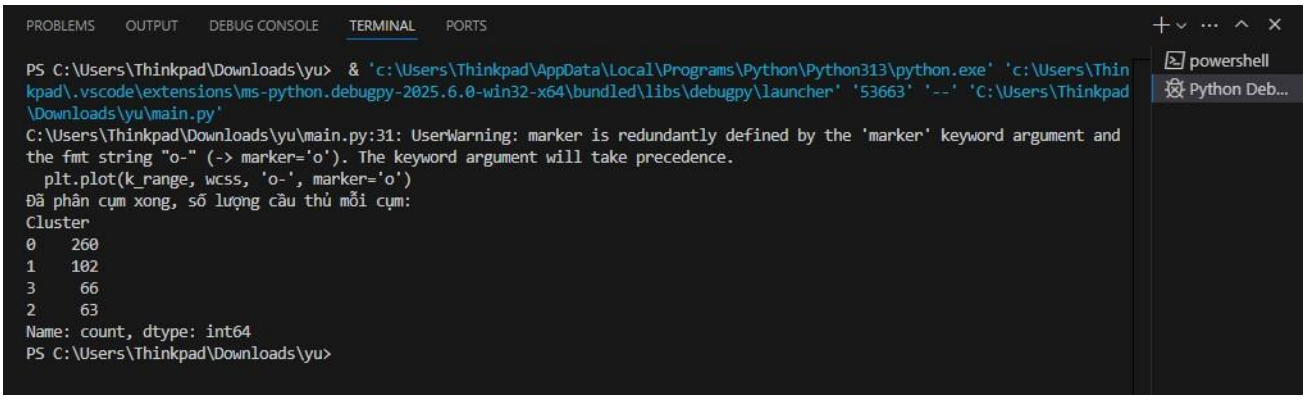
- Đầu tiên, áp dụng Elbow Method để xác định số lượng cụm phù hợp.
- Kết quả biểu đồ cho thấy:
  + Từ k=1 tới k=4, WCSS giảm mạnh.
  + Sau k=4, độ giảm WCSS bắt đầu chậm lại, đường cong trở nên "mượt" hơn.
  + Điểm gãy (elbow) xuất hiện rõ ở k = 4 → Do đó, **k=4** được chọn làm số lượng cụm tối ưu.

→ Kết luận: Chọn 4 cụm (k=4) là lựa chọn hợp lý nhất, cân bằng giữa việc tối ưu nội bộ cụm và tránh phân cụm quá nhỏ



2. Phân cụm cầu thủ bằng KMeans

- Sau khi xác định k=4, áp dụng thuật toán KMeans clustering lên dữ liệu đã chuẩn hóa.
- Kết quả phân cụm:



- File result_with_clusters.csv sẽ cho biết các cầu thủ ở cụm nào

→ Ý nghĩa:

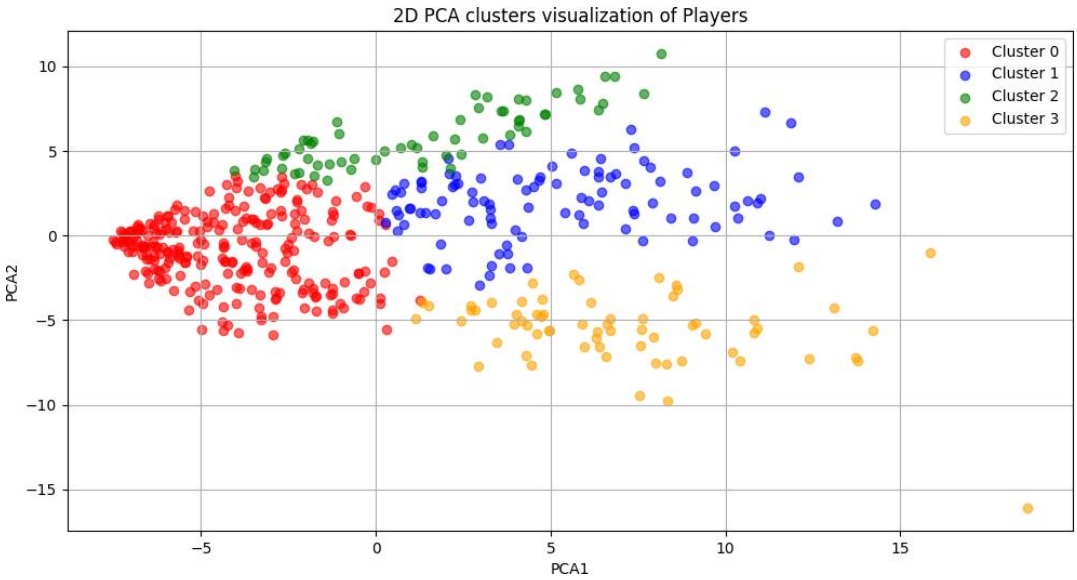
+ Cluster 0 là cụm lớn nhất, chiếm hơn nửa tổng số cầu thủ.

+ Cluster 2 và 3 có số lượng nhỏ hơn, đại diện cho các nhóm kỹ năng đặc biệt hơn.

3. Diễn giải ý nghĩa các cụm

Cluster	Đặc điểm chính
Cluster 0	Số bàn thắng (Goals), Sút chính xác (Shots on Target) cao.
Cluster 1	Số kiến tạo (Assists), Đường chuyền quyết định (Key Passes) cao.
Cluster 2	Tắc bóng (Tackles), Phá bóng (Clearances) vượt trội.
Cluster 3	Khả năng cắt bóng (Interceptions) + chuyền bóng (Pass Completion) tốt.

4. Trực quan hóa cụm bằng PCA 2D



- Ánh xạ dữ liệu từ không gian nhiều chiều xuống 2 chiều bằng phương pháp PCA.
- Vẽ biểu đồ scatter plot, mỗi điểm đại diện cho một cầu thủ.
- Nhạn xét biểu đồ:
  + Các cụm màu đỏ, xanh dương, xanh lá, cam có sự phân tách rõ ràng.
  + Hầu như không có hiện tượng chồng lấn nhiều giữa các cụm → Kết quả phân cụm rất khả quan.
  + Các cụm phân bổ hợp lý trên trục PCA1-PCA2.

→ Kết luận:

  + PCA đã hỗ trợ xác nhận rằng mô hình KMeans phân cụm hiệu quả, dễ dàng phân biệt các nhóm cầu thủ khác nhau.
  + Việc phân cụm này hoàn toàn có thể áp dụng trong thực tế cho tuyển chọn cầu thủ theo vai trò.