HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN LẬP TRÌNH VỚI PYTHON

BÀI TẬP LỚN MÔN HỌC
LẬP TRÌNH VỚI PYTHON

Họ và tên: Lê Văn Hà

Mã sinh viên: B22DCCN255

Nhóm lớp học : D22-117

Giảng viên giảng dạy: Thầy Kim Ngọc Bách

Hà Nội – 2024

BÀI 1 :

Viết chương trình Python thu thập dữ liệu phân tích cầu thủ với yêu cầu như sau:
-Thu thập dữ liệu thống kê [*] của tất cả các cầu thủ có số phút thi đấu nhiều hơn 90 phút tại giải bóng
đá ngoại hạng Anh mùa 2023-2024

1). Mục tiêu

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

2). Công cụ và thư viện sử dụng

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time

from player import Player
from player import Player_Manager
from player import Squad
from player import Squad_Manager

player_manager = Player_Manager()
squad_manager = Squad_Manager()
```

+) **Selenium**: Điều khiển trình duyệt để mở trang web, tìm kiếm và thao tác trên các thành phần HTML.

- Selenium tự động hóa trình duyệt thu thập dữ liệu (web scraping), và thực hiện các tác vụ lặp lại trên trang web.
- **Cài đặt Selenium**:

  pip install selenium

+) **Thời gian chờ (Time)**: Để đảm bảo dữ liệu tải đầy đủ trước khi xử lý.

+) Đồng thời khởi tạo các đối tượng quản lý cầu thủ và đội hình (Player_Manager, Squad_Manager).

3) Phân tích chi tiết từng phần

3.1)Hàm Kiểm Tra và Chuyển Đổi Dữ Liệu

```
def validdata(n):
    if n == '': return "N/a"
    return float(n)
```

Hàm validdata(n) nhận đầu vào là một giá trị. Nếu giá trị là chuỗi rỗng, nó trả về "N/a", nếu không, nó chuyển đổi giá trị thành kiểu float

3.2)Hàm Thu Thập Dữ Liệu Từ Web

```
def GetDataFromWeb(url, Xpath_player, Xpath_squad, Data_Name):
    driver = webdriver.Chrome()
    driver.get(url)

    resultPlayerData = []
    resultSquadData = []
```

+) Khởi tạo hàm GetDataFromWeb để thu thập dữ liệu từ một trang web.

+) Hàm này nhận các tham số như url, Xpath_player, Xpath_squad, và Data_Name. với :

param url: Địa chỉ URL của trang web cần thu thập dữ liệu.

param Xpath_player: Xpath của bảng cầu thủ.

param Xpath_squad: Xpath của bảng đội hình.

param Data_Name: Tên dữ liệu để hiển thị thông báo.

+) Khởi tạo trình duyệt Chrome và mở URL đã cho.

+) Tạo hai danh sách rỗng (resultPlayerData và resultSquadData) để lưu trữ dữ liệu.

3.3) Thu Thập Dữ Liệu Cầu Thủ

```
try:
    time.sleep(10)
    table2 = driver.find_element(By.XPATH, Xpath_player)
    rows2 = table2.find_elements(By.TAG_NAME, 'tr')

    for row in rows2:  # Bỏ qua hàng tiêu đề
        cols = row.find_elements(By.TAG_NAME, '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: resultPlayerData.append(data)
```

+) Trong khối try, chương trình chờ 10 giây để đảm bảo trang đã tải xong.

+) Tìm bảng cầu thủ bằng XPath và thu thập dữ liệu từ từng hàng (trừ hàng tiêu đề).

+) Xử lý các cột dữ liệu, bao gồm tên cầu thủ và các giá trị số, lưu trữ chúng vào danh sách resultPlayerData.

3.4) Đóng Trình Duyệt và Trả Về Dữ Liệu

```
    finally:
        driver.quit()
        print("Finish Page " + DataName)
    return resultPlayerData, resultSquadData
```

+) finally, đảm bảo rằng trình duyệt được đóng lại sau khi hoàn thành thu thập dữ liệu.

+) In thông báo hoàn thành với tên dữ liệu đã cho.

+) Trả về hai danh sách chứa dữ liệu cầu thủ và đội hình.

```
from tieu_de import header, row
from tieu_de import header2, row2
```

Nhập các biến header, row, header2, row2 từ mô-đun tieu_de. Đây có thể là các thông tin tiêu đề và hàm để lấy dữ liệu cho các cầu thủ và đội hình

3.5)Lưu Dữ Liệu Cầu Thủ vào File CSV

```
# Lưu dữ liệu của player_manager vào file CSV
with open('D:\Lê Văn Hà B22DCCN255\Thư mục code\result.csv', 'w', encoding='utf-8') as file:
    # Ghi tiêu đề (header) vào file
    file.write(",".join(header) + "\n")  # Sử dụng dấu phẩy thay vì dấu tab để lưu dưới dạng CSV

    # Ghi dữ liệu của các cầu thủ
    for player in player_manager.list_player:
        r = row(player)
        file.write(",".join(map(str, r)) + "\n")

print("Exam 1 Success - Player Data Saved in CSV file")
```

+)Ghi tiêu đề từ header vào file CSV, sử dụng dấu phẩy để phân tách các giá trị.

+) Lặp qua danh sách các cầu thủ (player_manager.list_player).

+) Sử dụng hàm row(player) để lấy dữ liệu của từng cầu thủ dưới dạng danh sách.

+) Chuyển đổi các giá trị trong danh sách thành chuỗi và nối chúng lại với nhau bằng dấu phẩy để ghi vào file.

+) In ra thông báo xác nhận rằng dữ liệu cầu thủ đã được lưu thành công vào file CSV.

3.6) Lưu Dữ Liệu Cầu Thủ vào File

```
with open('D:\Lê Văn Hà B22DCCN255\Thư mục code\squad_data.txt', 'w', encoding='utf-8') as file:
    # Ghi tiêu đề (header2) vào file
    file.write(",".join(header2) + "\n")

    # Ghi dữ liệu của các đội
    for squad in squad_manager.list_squad:
        r = row2(squad)
        file.write(",".join(map(str, r)) + "\n")

print("Exam 1 Success - Squad Data Saved in CSV file")
```

Thực hiện lấy dữ liệu của các đội bóng từ squad_manager và lưu vào file squad_data.txt. File kết quả sẽ chứa dữ liệu của từng đội bóng, được sắp xếp theo các cột tiêu đề đã xác định.

BÀI 2:

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

1) Đọc Dữ liệu từ File CSV:

```
df = pd.read_csv(r'D:\Lê Văn Hà B22DCCN255\Thư mục code\result.csv', header=2)
df.columns = df.columns.str.strip()
```

- *Sử dụng pd.read_csv() để đọc dữ liệu từ file results.csv và lưu vào DataFrame df.*
- *Loại bỏ khoảng trắng thừa trong tên cột bằng cách sử dụng df.columns.str.strip().*

2) Thay thế các giá trị không hợp lệ bằng 0

```
# Thay thế các giá trị "N/a" bằng 0
df.replace("N/a", 0, inplace=True)
```

Mục đích : để chuẩn hóa dữ liệu, giúp xử lý các giá trị không hợp lệ trong quá trình tính toán.

4) Kiểm Tra Tồn Tại Cột:

```
# Kiểm tra tên cột có tồn tại trong DataFrame không
for column in columns_to_check:
    if column not in df.columns:
        print(f"Cột '{column}' không tồn tại trong DataFrame.")
```

- *Sử dụng vòng lặp để kiểm tra xem mỗi cột trong columns_to_check có tồn tại trong DataFrame hay không. Nếu không tồn tại, in ra thông báo.*

5) Tìm Top 3 Cầu Thủ*:*

```
for column in columns_to_check:
    if column in df.columns:
        df[column] = pd.to_numeric(df[column], errors='coerce').fillna(0)

        # Lấy top 3 cao nhất
        top_3_highest[column] = df.nlargest(3, column)[[player_column, column]]

        # Lấy top 3 thấp nhất
        top_3_lowest[column] = df.nsmallest(3, column)[[player_column, column]]
```

- *Dùng từ điển top_3_highest và top_3_lowest để lưu trữ top 3 cầu thủ có điểm số cao nhất và thấp nhất cho từng chỉ số.*
- *Trong vòng lặp qua columns_to_check, nếu cột tồn tại, chuyển đổi giá trị của cột sang kiểu số (int hoặc float) và loại bỏ các hàng có giá trị NaN.*

- *Sử dụng nlargest(3, column) để tìm top 3 cầu thủ cao nhất và nsmallest(3, column) để tìm top 3 cầu thủ thấp nhất.*

6) In Kết Quả:

- *In tên cầu thủ của top 3 có điểm cao nhất và thấp nhất cho từng chỉ số.*

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

1) Đọc dữ liệu CSV*:*

```
df = pd.read_csv(r'D:\Lê Văn Hà B22DCCN255\Thư mục code\result.csv', header=[0, 1, 2])

for col_index in range(4, df.shape[1]):
```

- *Sử dụng pd.read_csv() để đọc dữ liệu từ tệp result.csv với header đa tầng (gồm ba hàng đầu tiên), nhằm dễ dàng phân biệt và xử lý các thuộc tính trong quá trình phân tích.*
- *Xác định các cột từ thứ 6 trở đi (index từ 4) chứa các thuộc tính cần tính toán.*

2) Tính toán thống kê cho toàn giải:

```
for col_index in range(4, df.shape[1]):
    column = df.iloc[:, col_index]
    column_numeric = pd.to_numeric(column, errors='coerce')

    # 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)
```

- *Với mỗi thuộc tính, chuyển dữ liệu của cột sang dạng số bằng pd.to_numeric() , Tham số errors='coerce' sẽ thay thế các giá trị không thể chuyển đổi bằng NaN*
- *Tính median, mean và độ lệch chuẩn của từng thuộc tính cho toàn giải, lưu trữ kết quả trong danh sách final_results.*

```
# Tạo một DataFrame để lưu kết quả cuối cùng
final_results = []
```

3) Tính toán thống kê cho từng đội:

```
# Tính toán cho từng đội
for team_name, team_data in df.groupby(('Unnamed: 2_level_0', 'Unnamed: 2_level_1', 'Team')):
    team_column = pd.to_numeric(team_data.iloc[:, col_index], errors='coerce')
    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])
```

- *Sử dụng groupby() để nhóm dữ liệu theo tên đội.*
- *Với mỗi đội và mỗi thuộc tính, tính toán median, mean và std tương tự như ở bước toàn giải.*
- *Kiểm tra nếu tên đội đã tồn tại trong final_results, cập nhật kết quả vào hàng tương ứng; nếu chưa có, thêm hàng mới cho đội đó.*

4) Tạo DataFrame kết quả:

```
final_df = pd.DataFrame(final_results)
```

- *Chuyển đổi danh sách final_results thành một DataFrame Pandas để dễ quản lý và ghi ra file.*

```
column_titles = ['Team']
for col_index in range(4, df.shape[1]):
    column_titles.extend([f'Median of {df.columns[col_index][2]}',
                          f'Mean of {df.columns[col_index][2]}',
                          f'Std of {df.columns[col_index][2]}'])

final_df.columns = column_titles
```

- *Cập nhật tiêu đề cột bao gồm tên các chỉ số cho từng thuộc tính (ví dụ: 'Median of Speed').*

5) Xuất kết quả:

- *Ghi kết quả ra file result2.csv để lưu trữ*

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

1. Đọc dữ liệu

```
# Đường dẫn tới tệp CSV
csv_file_path = r'D:\Lê Văn Hà B22DCCN255\Thư mục code\result2.csv'
```

- *Sử dụng thư viện pandas để đọc tệp CSV với header đa tầng, giúp dễ dàng quản lý các thuộc tính dữ liệu.*

2. Xác định các chỉ số cần phân tích

```
# Lấy tên các chỉ số từ cột thứ 5 trở đi (index 4)
attributes = df.columns[4:]
```

- *Lấy tên các chỉ số từ cột thứ 5 trở đi để phục vụ cho việc phân tích.*

3. Vẽ biểu đồ histogram cho toàn giải

```
for attribute in attributes:
    plt.figure(figsize=(10, 6))

    data = pd.to_numeric(df[attribute], errors='coerce').dropna()

    # Vẽ histogram với đường phân phối KDE
    sns.histplot(data, kde=True, bins=20, color='skyblue')

    plt.title(f'Histogram of {attribute[2]} for All Players')
    plt.xlabel(attribute[2])
    plt.ylabel('Frequency')

    # Lưu biểu đồ vào thư mục
    output_path = os.path.join(output_folder, f'{attribute[2]}_All.png')
    print(f'Saving histogram for all players to: {output_path}')
    plt.savefig(output_path)
    plt.close()
```

- *Sử dụng thư viện matplotlib và seaborn để vẽ histogram cho từng chỉ số, hiển thị sự phân bố dữ liệu với đường phân phối KDE*

```
# Tạo thư mục cho các biểu đồ nếu cần
output_folder = r'D:\Lê Văn Hà B22DCCN255\Thư mục hình vẽ'
os.makedirs(output_folder, exist_ok=True)
```

```
output_path = os.path.join(output_folder, f'{attribute[2]}_All.png')
print(f'Saving histogram for all players to: {output_path}')
plt.savefig(output_path)
plt.close()
```

Tạo và lưu biểu đồ vào thư mục output_folder

2d)

**) Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số*

```
import csv
max_stats = {}
```

Biến max_stats được khởi tạo dưới dạng một từ điển (dictionary). Biến này sẽ lưu trữ thông tin về đội bóng có giá trị cao nhất cho mỗi loại thống kê

```
with open('D:\\Slide 28 tech\\BTL\\BTL\\bai1\\file\\squad_data.txt', 'r', encoding='utf-8') as file:
    reader = csv.DictReader(file, delimiter='\t')
```

+) Tệp squad_data.txt được mở với chế độ đọc ('r'), và mã hóa là utf-8.

+) csv.DictReader được sử dụng để đọc tệp, với ký tự phân cách là tab ('\t'), nghĩa là các trường trong mỗi dòng được ngăn cách bằng tab.

```
    for row in reader:
        for stat, value in row.items():
            try:

                value = float(value)

                if stat not in max_stats:
                    max_stats[stat] = (row['name'], value)
                else:

                    if value > max_stats[stat][1]:
                        max_stats[stat] = (row['name'], value)
            except ValueError:

                continue
```

+) Đối với mỗi dòng (row) trong tệp, vòng lặp for sẽ lặp qua từng cặp (thống kê, giá trị).

+) Thử chuyển đổi value sang kiểu float. Nếu thành công:

- *Nếu thống kê (stat) chưa có trong max_stats, thêm vào với giá trị của name (tên cầu thủ/đội bóng) và value (giá trị thống kê).*
- *Nếu thống kê đã có, so sánh giá trị hiện tại với giá trị đã lưu. Nếu giá trị hiện tại lớn hơn, cập nhật lại thông tin trong max_stats.*

+) Nếu không thể chuyển đổi giá trị sang float, sẽ bỏ qua (tiếp tục với dòng tiếp theo).

```
for stat, (team, value) in max_stats.items():
    print(f"Đội bóng có {stat} cao nhất là {team} với giá trị {value}")
```

Cuối cùng, một vòng lặp khác sẽ lặp qua từng thống kê và in ra tên đội bóng (hoặc cầu thủ) có giá trị cao nhất cho từng loại thống kê cùng với giá trị đó.

Kết quả:

Kết quả thu được sẽ có fomat như sau :

```
Đội bóng có numberOfPlayer cao nhất là Sheffield Utd với giá trị 35.0
Đội bóng có Poss cao nhất là Manchester City với giá trị 65.2
Đội bóng có age cao nhất là Fulham với giá trị 28.4
Đội bóng có matches_played cao nhất là Arsenal với giá trị 38.0
Đội bóng có starts cao nhất là Arsenal với giá trị 418.0
Đội bóng có minutes cao nhất là Arsenal với giá trị 3420.0
Đội bóng có non_penalty_goals cao nhất là Manchester City với giá trị 85.0
Đội bóng có penalty_goals cao nhất là Chelsea với giá trị 11.0
Đội bóng có assists cao nhất là Manchester City với giá trị 69.0
Đội bóng có yellow_cards cao nhất là Chelsea với giá trị 109.0
Đội bóng có red_cards cao nhất là Burnley với giá trị 7.0
Đội bóng có xG cao nhất là Liverpool với giá trị 87.8
Đội bóng có npxG cao nhất là Liverpool với giá trị 80.4
Đội bóng có xAG cao nhất là Liverpool với giá trị 63.7
Đội bóng có PrgC cao nhất là Manchester City với giá trị 1127.0
Đội bóng có PrgP cao nhất là Liverpool với giá trị 2110.0
Đội bóng có per90_Gls cao nhất là Manchester City với giá trị 2.47
Đội bóng có per90_Ast cao nhất là Manchester City với giá trị 1.82
Đội bóng có per90_G+A cao nhất là Manchester City với giá trị 4.29
Đội bóng có per90_G-PK cao nhất là Manchester City với giá trị 2.24
Đội bóng có per90_G+A-PK cao nhất là Manchester City với giá trị 4.05
Đội bóng có per90_xG cao nhất là Liverpool với giá trị 2.31
Đội bóng có per90_xAG cao nhất là Liverpool với giá trị 1.68
```

**)Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024*

1)Thêm các thư viện cần thiết

```
import pandas as pd
from collections import Counter
```

+) pandas: Thư viện rất mạnh để xử lý dữ liệu dạng bảng, cho phép đọc, ghi và phân tích dữ liệu một cách dễ dàng.

+) Counter: Từ thư viện collections, cho phép đếm các đối tượng trong một iterable (trong trường hợp này là danh sách các đội bóng).

2) Nguồn dữ liệu

```
file_path = 'D:/Slide 28 tech/BTL/BTL/bai1/file/squad_data.txt'
```

3)Đếm Số Lần Xuất Hiện Của Mỗi Đội

```
team_counts = Counter(top_teams)
```

Sử dụng Counter để đếm số lần xuất hiện của từng đội bóng trong danh sách top_teams. Điều này sẽ cho biết đội nào có số lượng thống kê tốt nhất.

4)Tìm Đội Tốt Nhất

```
best_team = team_counts.most_common(1)[0]
```

Sử dụng phương thức most_common(1) của Counter để tìm đội bóng có số lượng thống kê tốt nhất. Kết quả trả về là một tuple chứa tên đội và số lượng thống kê

Kết quả:

Kết quả thu được sẽ có dạng

```
Top team in each category:
Manchester City: 6 categories
Newcastle Utd: 2 categories
Arsenal: 3 categories
Manchester Utd: 1 categories
Crystal Palace: 1 categories
Brighton: 1 categories

The team with the best overall performance is: Manchester City with 6 top categories
```

Bài 3:

Bai3a) Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau.

1)Đọc dữ liệu:

```
df = pd.read_csv('D:/Lê Văn Hà B22DCCN255/Thư mục code/result.csv', header=2)
```

Đoạn mã đọc dữ liệu từ tệp CSV và chọn các cột từ cột thứ 5 trở đi để thực hiện phân tích. Cột header=2 chỉ định rằng tiêu đề của tệp CSV nằm ở dòng thứ 3.

2) Tiền xử lý dữ liệu

```
data.dropna(axis='columns', inplace=True)
data = data.apply(pd.to_numeric, errors='coerce').fillna(0)
```

+) Dữ liệu không có giá trị (NaN) được loại bỏ.

+) Các giá trị không hợp lệ được thay thế bằng 0.

3)Chuẩn hóa dữ liệu

```
data = ((data - data.min()) / (data.max() - data.min())) * 9 + 1
```

Dữ liệu được chuẩn hóa về khoảng [1, 10] để đảm bảo rằng tất cả các đặc trưng đều có cùng tỉ lệ.

4) Giảm chiều dữ liệu

```
pca = PCA(n_components=2)
data_2d = pca.fit_transform(data)
```

+) Sử dụng PCA để giảm chiều dữ liệu xuống còn 2 chiều, giúp dễ dàng trực quan hóa dữ liệu.

5) Tối ưu hóa K-Means

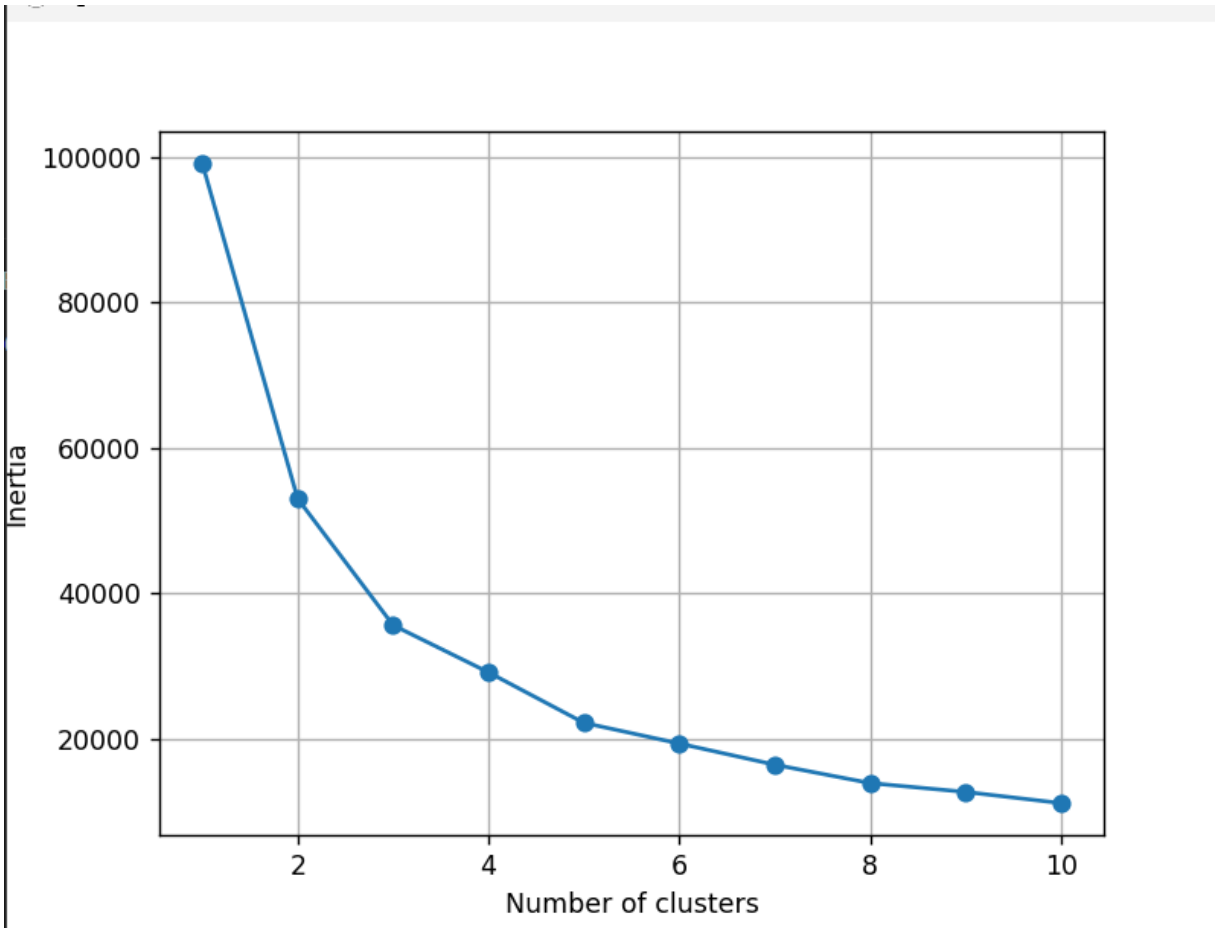
```
def optimise_k_means(data_2d, max_k):
    means = []
    inertias = []

    for k in range(1, max_k + 1):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data_2d)

        means.append(k)
        inertias.append(kmeans.inertia_)

    # Vẽ biểu đồ
    plt.plot(means, inertias, 'o-')
    plt.xlabel("Number of clusters")
    plt.ylabel("Inertia")
    plt.grid(True)
    plt.savefig(r"D:\Lê Văn Hà B22DCCN255\Thư mục hình vẽ\inertia_plot.png")
    plt.show()
```

+) Hàm optimise_k_means chạy thuật toán K-Means với các giá trị k từ 1 đến max_k.

+) Inertia (một chỉ số đánh giá mức độ phân cụm) được tính toán cho từng giá trị k và lưu trữ để vẽ biểu đồ.

+) Biểu đồ được vẽ để hiển thị mối quan hệ giữa số lượng cụm và inertia, giúp tìm ra số lượng cụm tối ưu thông qua phương pháp elbow.

Kết quả thu được



Bai3b)

Qua đó có thể thấy được điểm gãy tại 3 và từ đó có thể xác định được k = 3 là tối ưu nhất cho các cầu thủ này

Bai3c) Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D

1) Đọc và xử lý dữ liệu:

```
df = pd.read_csv('D:/Lê Văn Hà B22DCCN255/Thư mục code/result.csv', header=2)
data = df[df.columns[4:]].copy()
data = data.apply(pd.to_numeric, errors='coerce').fillna(0)
data.dropna(axis='columns', inplace=True)
```

+) Đoạn mã đọc dữ liệu từ tệp CSV và chọn các cột từ cột thứ 5 trở đi để phân tích.

+) Dữ liệu không hợp lệ được chuyển thành NaN và thay thế bằng 0. Các cột không có giá trị (NaN) cũng bị loại bỏ.

2)Chuẩn hóa dữ liệu

```
data = ((data - data.min()) / (data.max() - data.min())) * 9 + 1
```

Dữ liệu được chuẩn hóa về khoảng [1, 10] để đảm bảo tất cả các đặc trưng có cùng tỉ lệ

3)Khởi tạo tâm ngẫu nhiên:

```
def random_centroids(data, k):
    centroids = []
    for _ in range(k):
        centroid = data.apply(lambda x: float(x.sample().iloc[0]))
        centroids.append(centroid)
    return pd.concat(centroids, axis=1)
```

khởi tạo k tâm ngẫu nhiên từ dữ liệu đầu vào. Mỗi centroid được chọn ngẫu nhiên từ các điểm dữ liệu.

4) Xác định nhãn cho từng điểm dữ liệu

```
def get_labels(data, centroids):
    distances = centroids.apply(lambda x: np.sqrt(((data - x) ** 2).sum(axis=1)))
    return distances.idxmin(axis=1)
```

tính khoảng cách giữa từng điểm dữ liệu và các centroid, gán nhãn cho mỗi điểm dữ liệu dựa trên centroid gần nhất

5) Tính toán centroid mới

```
def new_centroids(data, labels, k):
    centroids = data.groupby(labels).apply(lambda x: np.exp(np.log(x).mean())).T
    return centroids
```

Centroid mới được tính toán dựa trên giá trị trung bình của các điểm dữ liệu trong mỗi cụm

6) Vẽ biểu đồ

```
def plot_clusters(data, labels, centroids, iteration):
    pca = PCA(n_components=2)
    data_2d = pca.fit_transform(data)
    centroids_2d = pca.transform(centroids.T)
    time.sleep(1)
    clear_output(wait=True)
    plt.title(f'Iteration {iteration}')
    plt.scatter(x=data_2d[:,0], y=data_2d[:,1], c=labels)
    plt.scatter(x=centroids_2d[:,0], y=centroids_2d[:,1])
    plt.savefig(r"D:\Lê Văn Hà B22DCCN255\Thư mục hình vẽ\kmeans_pca.png")
    plt.show()
```

Kết quả thu được