

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Bài Báo Cáo

Môn: Lập Trình Python

Giảng Viên: Kim Ngọc Bách

Sinh Viên : Phạm Quốc Anh

MSV: B22DCCN040

Email : anhpham030704@gmail.com

Nhóm học phần : 11

Bài 1:

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

- **requests**: Thư viện dùng để gửi các yêu cầu HTTP đến website và nhận dữ liệu trả về.
- **BeautifulSoup**: Thư viện dùng để phân tích và trích xuất dữ liệu từ trang web HTML.
- **pandas**: Thư viện mạnh mẽ dùng để xử lý và phân tích dữ liệu trong Python, đặc biệt là làm việc với dữ liệu dạng bảng.

```
URL = "https://fbref.com/en/comps/9/Premier-League-Stats"
```

URL: Địa chỉ của trang web FBref chứa bảng thống kê cầu thủ của giải Ngoại hạng Anh.

```
def get_player_data():
    response = requests.get(URL)
    if response.status_code != 200:
        print("Failed to fetch data from FBref.")
        return None
```

Gửi yêu cầu GET tới URL để lấy dữ liệu từ trang web FBref.

Kiểm tra mã trạng thái HTTP trả về. Nếu không phải mã 200 (tức là yêu cầu thành công), in thông báo lỗi và trả về None.

```
soup = BeautifulSoup(response.text, 'html.parser')
table = soup.find("table", {"id": "stats_standard_9"})
if not table:
    print("Table not found on the page.")
    return None
```

Dùng BeautifulSoup để phân tích nội dung HTML của trang web đã được tải về (dưới dạng văn bản).

Tìm bảng có **id="stats_standard_9"**, là bảng chứa thống kê cầu thủ giải Ngoại hạng Anh.

Nếu không tìm thấy bảng trên trang, in thông báo lỗi và trả về None.

```
headers = [th.text.strip() for th in table.find("thead").find_all("th")]
rows = []
for row in table.find("tbody").find_all("tr"):
    if row.find("th", {"scope": "row"}):
        data = [cell.text.strip() if cell.text.strip() else "N/A" for cell in row.find_all("td")]
        rows.append(data)
```

Khởi tạo một danh sách rỗng `rows` để chứa dữ liệu của các cầu thủ.

Duyệt qua tất cả các dòng trong phần `<tbody>` của bảng:

- Nếu dòng chứa một thẻ `<th>` với thuộc tính `scope="row"` (dùng để đánh dấu các dòng chứa dữ liệu của cầu thủ), ta sẽ trích xuất tất cả các giá trị của các ô dữ liệu trong dòng.
- Dùng `strip()` để loại bỏ khoảng trắng thừa và nếu ô dữ liệu trống, thay thế bằng "N/A".

```
df = pd.DataFrame(rows, columns=headers[1:])
df['Player'] = df['Player'].str.split('\\').str[0]
df = df[df['Min'].replace("N/A", "0").astype(int) > 90]
df = df.sort_values(by=['Player', 'Age'], ascending=[True, True])
df.to_csv("results.csv", index=False)
return df
```

Chuyển dữ liệu thu thập được (dưới dạng danh sách `rows`) thành một `DataFrame` của `pandas` và gán các tên cột từ danh sách `headers`. `headers[1:]` bỏ qua phần tử đầu tiên (do nó là "Player", đã được tách riêng ở bước trước).


Tách tên cầu thủ nếu nó có chứa ký tự `\` và chỉ giữ lại phần tên trước ký tự đó (trong trường hợp có sự phân cách giữa tên và biệt danh).


Lọc các cầu thủ có số phút thi đấu (Min) lớn hơn 90. Trước khi lọc, ta thay thế giá trị "N/A" bằng 0 và chuyển đổi các giá trị của cột Min thành kiểu số nguyên.

Bài 2 ;

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("results.csv")
attributes = ['Min', 'Gls', 'Ast', 'xG', 'xA', 'Shots', 'Cmp%', 'Tkl', 'Int', 'PrgC']
top_bottom = {}
```

 **df = pd.read_csv("results.csv")**: Đọc dữ liệu từ file `results.csv` vào `DataFrame` `df`.

 **attributes**: Danh sách các chỉ số (cột dữ liệu) mà bạn sẽ phân tích cho mỗi cầu thủ.

🎬 **top_bottom**: Khởi tạo một từ điển để lưu trữ top 3 cầu thủ có điểm cao nhất và thấp nhất cho từng chỉ số.

```
for attr in attributes:
    df[attr] = pd.to_numeric(df[attr], errors='coerce')
    top_bottom[attr] = {
        'Top 3': df.nlargest(3, attr)[['Player', attr]],
        'Bottom 3': df.nsmallest(3, attr)[['Player', attr]]
    }

results = []
```

- **df[attr] = pd.to_numeric(df[attr], errors='coerce')**: Chuyển các giá trị của mỗi chỉ số về kiểu số (numeric). Nếu có giá trị không thể chuyển đổi, thay thế bằng NaN (coerce).

- 🎬 **df.nlargest(3, attr)**: Tìm 3 cầu thủ có giá trị cao nhất cho chỉ số attr.

- 🎬 **df.nsmallest(3, attr)**: Tìm 3 cầu thủ có giá trị thấp nhất cho chỉ số attr.

- 🎬 Lưu kết quả vào từ điển top_bottom.

```
for attr in attributes:
    median = df[attr].median()
    mean = df[attr].mean()
    std = df[attr].std()
    results.append({'Attribute': attr, 'Type': 'All', 'Median': median, 'Mean': mean, 'Std': std})
    for team, group in df.groupby('Team'):
        median_team = group[attr].median()
        mean_team = group[attr].mean()
        std_team = group[attr].std()
        results.append({'Attribute': attr, 'Type': team, 'Median': median_team, 'Mean': mean_team, 'Std': std_team})
```

- **df[attr].median()**: Tính giá trị trung vị (median) của chỉ số attr.

- 🎬 **df[attr].mean()**: Tính giá trị trung bình (mean) của chỉ số attr.

- 🎬 **df[attr].std()**: Tính độ lệch chuẩn (standard deviation) của chỉ số attr.

- 🎬 Sau khi tính toán cho toàn bộ dữ liệu ('All'), tiếp tục tính toán các giá trị này cho từng đội trong giải Ngoại hạng Anh.

- 🎬 Tất cả các kết quả được lưu vào một DataFrame results_df và xuất ra file CSV results2.csv.

```
for attr in attributes:
    plt.hist(df[attr].dropna(), bins=20, alpha=0.7, label='All')
    for team, group in df.groupby('Team'):
        plt.hist(group[attr].dropna(), bins=20, alpha=0.5, label=team)
    plt.title(f'Histogram of {attr}')
    plt.xlabel(attr)
    plt.ylabel('Frequency')
    plt.legend(loc='upper right')
    plt.show()
```

🎬 **plt.hist(df[attr].dropna(), bins=20, alpha=0.7, label='All')**: Vẽ histogram cho dữ liệu của toàn bộ cầu thủ. dropna() loại bỏ các giá trị NaN.

🎬 **for team, group in df.groupby('Team')**: Lặp qua từng đội để vẽ histogram của mỗi đội.

🎬 **plt.show()**: Hiện thị biểu đồ.

```
team_scores = df.groupby('Team')[attributes].sum()
best_teams = team_scores.idxmax()

print(best_teams)
```

- **df.groupby('Team')[attributes].sum()**: Tính tổng điểm của mỗi đội cho tất cả các chỉ số.
- **team_scores.idxmax()**: Tìm đội có tổng điểm cao nhất cho mỗi chỉ số.
- **print(best_teams)**: In ra đội có chỉ số cao nhất cho từng thuộc tính.

Bài 3 :

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from math import pi
import argparse
```

• **pandas**: Dùng để xử lý và phân tích dữ liệu dưới dạng bảng.

🎬 **KMeans**: Dùng để thực hiện thuật toán phân cụm K-means.

🎬 **PCA**: Dùng để giảm số chiều dữ liệu xuống 2 chiều (principal component analysis).

🎬 **matplotlib.pyplot**: Dùng để vẽ biểu đồ.

🎬 **StandardScaler**: Dùng để chuẩn hóa dữ liệu (scaling).

🎬 **pi**: Để sử dụng hằng số π khi vẽ biểu đồ radar.

🎬 **argparse**: Dùng để phân tích các đối số dòng lệnh.


```
def perform_kmeans(df, columns_to_cluster, num_clusters=4):
    data = df[columns_to_cluster].replace("N/A", 0).astype(float)
    scaler = StandardScaler()
    data_scaled = scaler.fit_transform(data)

    kmeans = KMeans(n_clusters=num_clusters)
    df['Cluster'] = kmeans.fit_predict(data_scaled)

    pca = PCA(n_components=2)
    data_pca = pca.fit_transform(data_scaled)
    df['PCA1'] = data_pca[:, 0]
    df['PCA2'] = data_pca[:, 1]

    plt.figure(figsize=(10, 8))
    for cluster in range(num_clusters):
        cluster_data = df[df['Cluster'] == cluster]
        plt.scatter(cluster_data['PCA1'], cluster_data['PCA2'], label=f'Cluster {cluster}')
    plt.xlabel('PCA1')
    plt.ylabel('PCA2')
    plt.legend()
    plt.title('K-means Clustering with PCA')
    plt.show()
    return df
```

- **data = df[columns_to_cluster].replace("N/A", 0).astype(float):** Lấy dữ liệu từ các chỉ số được chọn (cột trong columns_to_cluster), thay thế "N/A" bằng 0 và chuyển dữ liệu thành kiểu số thực.

- **scaler = StandardScaler():** Tạo một đối tượng chuẩn hóa dữ liệu.

- **data_scaled = scaler.fit_transform(data):** Áp dụng chuẩn hóa cho dữ liệu.

- **kmeans = KMeans(n_clusters=num_clusters):** Khởi tạo thuật toán K-means với số lượng cụm là num_clusters.

- **df['Cluster'] = kmeans.fit_predict(data_scaled):** Thực hiện phân cụm và gán nhãn cụm vào cột 'Cluster' của DataFrame.

- **pca = PCA(n_components=2):** Khởi tạo PCA để giảm số chiều dữ liệu xuống còn 2.

- **data_pca = pca.fit_transform(data_scaled):** Thực hiện PCA trên dữ liệu đã chuẩn hóa.

- **df['PCA1'] = data_pca[:, 0] và df['PCA2'] = data_pca[:, 1]:** Lưu kết quả PCA vào các cột mới trong DataFrame (PCA1 và PCA2).

- **Vẽ biểu đồ phân cụm trên mặt phẳng 2D sử dụng PCA1 và PCA2.**

```
def plot_radar(df, player1, player2, attributes):
    attributes = attributes.split(",")
    data = df[df['Player'].isin([player1, player2])]
    data = data.set_index('Player')[attributes].astype(float)

    values1 = data.loc[player1].values.flatten().tolist()
    values2 = data.loc[player2].values.flatten().tolist()
    values1 += values1[:1]
    values2 += values2[:1]

    angles = [n / float(len(attributes)) * 2 * pi for n in range(len(attributes))]
    angles += angles[:1]

    fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
    ax.plot(angles, values1, linewidth=1, linestyle='solid', label=player1)
    ax.fill(angles, values1, alpha=0.3)
    ax.plot(angles, values2, linewidth=1, linestyle='solid', label=player2)
    ax.fill(angles, values2, alpha=0.3)

    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(attributes)
    plt.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))
    plt.show()
```

- **attributes = attributes.split(",")**: Chuyển chuỗi các thuộc tính từ tham số dòng lệnh thành danh sách.

- **data = df[df['Player'].isin([player1, player2])]**: Lọc dữ liệu chỉ giữ lại hai cầu thủ đã được chọn.

- **values1 và values2**: Lấy giá trị của các chỉ số cho hai cầu thủ đã chọn và bổ sung giá trị đầu tiên vào cuối danh sách để tạo thành một vòng khép kín cho biểu đồ radar.

- **angles**: Tính các góc để vẽ biểu đồ radar.

- **ax.plot và ax.fill**: Vẽ các đường và lấp đầy các khu vực của biểu đồ radar cho từng cầu thủ.

- **ax.set_xticks và ax.set_xticklabels**: Đặt các nhãn cho các trục của biểu đồ radar.

```
if __name__ == "__main__":
    df = pd.read_csv("results.csv")
    columns_to_cluster = ['Min', 'Gls', 'Ast', 'xG', 'xA', 'Shots', 'Cmp%', 'Tkl', 'Int', 'PrgC']
    df = perform_kmeans(df, columns_to_cluster)

    parser = argparse.ArgumentParser()
    parser.add_argument("--p1", required=True)
    parser.add_argument("--p2", required=True)
    parser.add_argument("--Attribute", required=True)
    args = parser.parse_args()
    plot_radar(df, args.p1, args.p2, args.Attribute)
```

- **df = pd.read_csv("results.csv")**: Đọc dữ liệu từ file CSV.

- **columns_to_cluster = [...]**: Định nghĩa danh sách các chỉ số cần dùng cho việc phân cụm.

- 🎬 **df = perform_kmeans(df, columns_to_cluster):** Thực hiện phân cụm K-means và PCA.
- 🎬 **argparse.ArgumentParser():** Tạo đối tượng phân tích các đối số dòng lệnh.
- 🎬 **parser.add_argument(...):** Định nghĩa các tham số dòng lệnh cần thiết: tên hai cầu thủ (--p1, --p2) và các chỉ số (--Attribute).
- 🎬 **args = parser.parse_args():** Phân tích các tham số dòng lệnh.
- 🎬 **plot_radar(df, args.p1, args.p2, args.Attribute):** Vẽ biểu đồ radar để so sánh hai cầu thủ theo các chỉ số đã chọn.