

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CƠ BẢN I

BỘ MÔN TIN HỌC CƠ SỞ



THỰC HÀNH TIN HỌC CƠ SỞ

Giảng viên hướng dẫn	: Kim Ngọc Bách
Họ và tên sinh viên	: Nguyễn Cảnh Đức Anh
Mã sinh viên	: B23DCCE006
Lớp	: CE06-CLC
Nhóm	: Cá nhân

Hà Nội – 2023

BÁO CÁO BÀI TẬP LỚN

Bài tập 1:

I. Thư Viện Sử Dụng

Chương trình sử dụng các thư viện Python tiêu chuẩn cho Web Scraping và xử lý dữ liệu:

1. **requests**: Thư viện dùng để thực hiện các yêu cầu HTTP (GET, POST,...) đến trang web. Nó được sử dụng để tải nội dung HTML của trang thống kê chính và các trang cá nhân của cầu thủ.
2. **BeautifulSoup (bs4)**: Thư viện dùng để phân tích cú pháp (parse) nội dung HTML đã tải về. Nó giúp dễ dàng tìm kiếm các phần tử HTML (như bảng, dòng, cột) dựa trên các thuộc tính (như **id**, **class**, **data-stat**).

3. **pandas**: Thư viện mạnh mẽ cho phân tích và thao tác dữ liệu. Dữ liệu thu thập được tổ chức thành cấu trúc DataFrame của pandas, giúp dễ dàng xử lý, sắp xếp và lưu vào tệp CSV.
4. **time**: Thư viện cung cấp các chức năng liên quan đến thời gian. Hàm **time.sleep()** được sử dụng để tạo độ trễ nhỏ giữa các yêu cầu tải trang, giúp giảm tải cho máy chủ của trang web và tránh bị chặn.

IV. Cấu Trúc Code và Logic

Chương trình được tổ chức thành các hàm để dễ quản lý và tái sử dụng:

1. Hàm **get_player_data_from_row(row)**:

- **Mục đích**: Trích xuất thông tin cơ bản của cầu thủ (Tên, Đội, Vị trí, Tuổi, Thời gian thi đấu, v.v.) và quan trọng nhất là **URL đến trang cá nhân của cầu thủ** từ một dòng (**<tr>**) trong bảng thống kê chính trên trang EPL tổng hợp.
- **Chi tiết**: Sử dụng **BeautifulSoup** để tìm các thẻ **<td>** hoặc **<th>** với thuộc tính **data-stat** tương ứng với từng thông tin cần thiết. Trích xuất văn bản từ các thẻ này. Đặc biệt, tìm thẻ **<a>** trong cột tên cầu thủ để lấy đường link (**href**) đến trang chi tiết của cầu thủ.
- **Xử lý lỗi**: Trả về **None** nếu có lỗi trong quá trình trích xuất một dòng, hoặc nếu không tìm thấy thông tin cần thiết, gán giá trị 'N/a'.

2. Hàm **Workspace_detailed_player_stats(player_url)**

- **Mục đích**: Truy cập vào URL trang cá nhân của một cầu thủ và trích xuất tất cả các thống kê chi tiết yêu cầu.
- **Chi tiết**:
 - Sử dụng **requests** để tải trang HTML của cầu thủ.
 - Sử dụng **BeautifulSoup** để phân tích cú pháp.

- **Hàm Helper `get_stat(soup, table_id, data_stat, is_per_90=False)`:**

Đây là một hàm nội bộ giúp tìm kiếm giá trị thống kê một cách hiệu quả hơn. Nó nhận đối tượng `soup` của trang, ID của bảng (`table_id`) và thuộc tính `data-stat` của cột cần tìm. FBref thường giấu các bảng thống kê nâng cao trong các comment HTML để tải nhanh hơn, nên hàm này cũng kiểm tra trong các comment đã được loại bỏ comment tag. Sau khi tìm được bảng, nó tìm dòng dữ liệu chính trong `<tbody>` (thường là dòng đầu tiên không phải header phụ) và trích xuất dữ liệu từ cột có `data-stat` phù hợp.

- Gọi hàm `get_stat` lặp đi lặp lại với các `table_id` và `data-stat` khác nhau để thu thập tất cả các thống kê chi tiết (Thủ môn, Sút bóng, Chuyển bóng, v.v.).

- Lưu trữ các thống kê này vào một từ điển.

3. Hàm `scrape_epl_2025_stats(url)`:

- **Mục đích:** Đây là hàm chính điều phối toàn bộ quá trình scraping.

- **Chi tiết:**

- **Bước 1: Thu thập thông tin cơ bản và URL:**

- Tải trang tổng hợp EPL 2024-2025.
- Tìm bảng thống kê chính (`stats_standard`).
- Lặp qua từng dòng cầu thủ trong `<tbody>`.
- Sử dụng `get_player_data_from_row` để lấy thông tin cơ bản và URL.
- Kiểm tra điều kiện "Thời gian thi đấu (Min) > 90". Nếu thỏa mãn, thêm thông tin cơ bản của cầu thủ vào danh sách `filtered_players_basic_info`.

- **Bước 2: Thu thập thống kê chi tiết:**

- Lặp qua danh sách `filtered_players_basic_info`.
- Đối với mỗi cầu thủ, lấy URL trang cá nhân.

- Gọi hàm `Workspace_detailed_player_stats` để lấy các thống kê chi tiết.
 - Thêm từ điển này vào danh sách `all_players_full_stats`.
 - Sử dụng `time.sleep(1)` để tạm dừng 1 giây giữa các lần tải trang cá nhân, tránh gửi quá nhiều yêu cầu liên tục.
 - **Bước 3: Tạo DataFrame, Sắp xếp và Lưu:**
 - Chuyển danh sách `all_players_full_stats` thành DataFrame của pandas.
 - Sắp xếp DataFrame theo cột 'Name' theo thứ tự tăng dần (`ascending=True`).
 - Lưu DataFrame đã sắp xếp vào tệp `results.csv` bằng phương thức `.to_csv(index=False)`.
4. Khối `if __name__ == "__main__":`
- Gọi hàm chính `scrape_epl_2025_stats` với URL này.
- In thông báo khi quá trình hoàn thành.

V. Giải Thích Các Biến và Chi Tiết Thống Kê

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

Bài tập 2:

1. Thư viện sử dụng

```
import pandas as pd      # Xử lý bảng dữ liệu dạng DataFrame

from bs4 import BeautifulSoup # Phân tích cú pháp HTML

import requests          # Gửi yêu cầu HTTP

import numpy as np       # Xử lý dữ liệu số (ở đây không dùng nhiều)
```

2. Mục đích hàm `fetch_player_stats(url)`

- Gửi HTTP request đến trang web chứa dữ liệu thống kê cầu thủ
- Phân tích cú pháp HTML để tìm bảng dữ liệu
- Lọc ra những cầu thủ có số phút chơi > 90
- Nếu có, truy cập vào trang chi tiết của từng cầu thủ để thu thập thêm chỉ số (ghi bàn, kiến tạo, chuyền bóng, phòng ngự...)
- Kết quả cuối cùng là một DataFrame chứa thống kê của các cầu thủ

3. Phân tích các bước chính trong hàm

a. Gửi request và lấy HTML

Đoạn code bạn đưa là một hàm Python để thu thập (scrape) và phân tích thống kê cầu thủ bóng đá từ trang [FBref.com](https://fbref.com). Dưới đây là phần phân tích chi tiết các biến, thư viện, và các điểm đáng chú ý, nhằm giúp người đọc dễ hiểu hơn:

1. Thư viện sử dụng

```
import pandas as pd      # Xử lý bảng dữ liệu dạng DataFrame
```

```
from bs4 import BeautifulSoup # Phân tích cú pháp HTML
```

```
import requests          # Gửi yêu cầu HTTP
```

```
import numpy as np       # Xử lý dữ liệu số (ở đây không dùng nhiều)
```

2. Mục đích hàm `fetch_player_stats(url)`

- **Gửi HTTP request đến trang web chứa dữ liệu thống kê cầu thủ**
- **Phân tích cú pháp HTML để tìm bảng dữ liệu**
- **Lọc ra những cầu thủ có số phút chơi > 90**
- **Nếu có, truy cập vào trang chi tiết của từng cầu thủ để thu thập thêm chỉ số (ghi bàn, kiến tạo, chuyền bóng, phòng ngự...)**
- **Kết quả cuối cùng là một DataFrame chứa thống kê của các cầu thủ**

3. Phân tích các bước chính trong hàm

a.

Gửi request và lấy HTML

```
headers = {'User-Agent': 'Mozilla/5.0'}
```

```
response = requests.get(url, headers=headers)
```

```
soup = BeautifulSoup(response.content, 'html.parser')
```

- **Giả lập trình duyệt để tránh bị chặn bot**
- **Phân tích cú pháp HTML từ trang web**

b.

Tìm bảng thống kê chính

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

- Tìm bảng có id="stats_standard" (thống kê tiêu chuẩn)

c.

Duyệt từng cầu thủ trong bảng

```
for row in tbody.find_all('tr'):
```

- Kiểm tra có phải là dòng dữ liệu hay không (<th scope="row">)
- Lấy các thông tin cơ bản:
 - Quốc tịch, Tên, Đội, Vị trí, Tuổi, Trận, Trận đá chính, Phút thi đấu
 - Nếu phút thi đấu > 90, sẽ lấy thêm các chỉ số nâng cao

d.

Chỉ số cơ bản cho cầu thủ có phút thi đấu > 90

```
player_stats['Gls'], player_stats['Ast'], ...
```

- Ghi bàn, kiến tạo, thẻ vàng, thẻ đỏ, xG, xAG, các chỉ số tiến công...

e.

Lấy chi tiết từ trang riêng của từng cầu thủ

```
player_link = row.find('a', string=player_stats['Name'])
```



```
player_url = f"https://fbref.com{player_link['href']}"
```

```
player_soup = BeautifulSoup(requests.get(player_url).content, 'html.parser')
```

- Truy cập vào trang riêng để lấy các bảng thống kê khác:
 - Thủ môn: stats_keeper (bị thủng lưới, phần trăm cứu thua...)
 - Sút bóng: stats_shooting
 - Chuyển bóng: stats_passing
 - Kiến tạo: stats_gca
 - Phòng ngự: stats_defense

4. Lưu ý

- Hàm chưa trả về gì (thiếu return pd.DataFrame(players_data))
- Có một lỗi ở dòng cuối:

```
player_stats['Lost'] = '
```

- => dòng này đang bị lỗi cú pháp và có thể khiến hàm không chạy.

5. Đề xuất cải thiện

- Thêm return ở cuối để trả kết quả:

```
return pd.DataFrame(players_data)
```

-

- Xử lý lỗi timeout hoặc lỗi mạng khi truy cập từng cầu thủ
- Sử dụng try-except cho từng phần nhỏ để không dừng cả hàm nếu 1 cầu thủ

Bài tập 3:

Đoạn code bạn chia sẻ thực hiện một quy trình phân cụm cầu thủ (hoặc bất kỳ thực thể nào trong results.csv) bằng thuật toán K-Means clustering, kết hợp với PCA để trực quan hóa dữ liệu. Dưới đây là phần giải thích chi tiết từng bước, giúp bạn hoặc người đọc dễ hiểu hơn:

Giải thích chi tiết đoạn code

Thư viện sử dụng

```
import pandas as pd # Xử lý dữ liệu dạng bảng
```

```
import matplotlib.pyplot as plt # Vẽ biểu đồ
```

```
from sklearn.preprocessing import StandardScaler # Chuẩn hóa dữ liệu
```

```
from sklearn.decomposition import PCA # Phân tích thành phần chính (giảm chiều dữ liệu)
```

```
from sklearn.cluster import KMeans # Thuật toán phân cụm KMeans
```

Bước 1: Đọc dữ liệu

```
df = pd.read_csv("results.csv")
```

- Đọc dữ liệu từ file CSV (results.csv) chứa thông tin thống kê cầu thủ.

Bước 2: Tiền xử lý dữ liệu

```
numeric_df = df.select_dtypes(include='number').dropna(axis=1)
```

- Chỉ giữ lại các biến dạng số học để phục vụ phân cụm.
- Loại bỏ các cột có giá trị thiếu (NaN).

```
scaler = StandardScaler()
```

```
scaled_data = scaler.fit_transform(numeric_df)
```

- Chuẩn hóa dữ liệu: Chuyển mỗi biến về thang đo có trung bình = 0 và độ lệch chuẩn = 1.
- Việc này giúp thuật toán KMeans hoạt động hiệu quả hơn.

Bước 3: Tìm số cụm tối ưu (Elbow Method)

```
inertia = []
```

```
K_range = range(1, 11)
```

```
for k in K_range:
```

```
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
```

```
kmeans.fit(scaled_data)
```

```
inertia.append(kmeans.inertia_)
```

- **Inertia:** Tổng sai số trong cụm (tổng khoảng cách từ mỗi điểm tới tâm cụm của nó).
- **Elbow Method:** Tìm giá trị k tại đó độ giảm của inertia bắt đầu chậm lại – gọi là “điểm khuỷu” (elbow).

```
plt.plot(K_range, inertia, 'bo-')
```

- Vẽ biểu đồ Elbow để giúp xác định số cụm tối ưu trực quan.

Bước 4: Phân cụm với số cụm tối ưu (giả sử là K=3)

```
optimal_k = 3
```

```
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init='auto')
```

```
labels = kmeans.fit_predict(scaled_data)
```

```
df["Cluster"] = labels
```

- Dùng KMeans để phân cụm dữ liệu đã chuẩn hóa.
- Gán nhãn cụm cho từng cầu thủ vào cột "Cluster" trong dataframe.

Bước 5: Giảm chiều dữ liệu với PCA (2 thành phần)

```
pca = PCA(n_components=2)
```

```
pca_data = pca.fit_transform(scaled_data)
```

```
df["PCA1"] = pca_data[:, 0]
```

```
df["PCA2"] = pca_data[:, 1]
```

- PCA giúp nén dữ liệu nhiều chiều thành 2 chiều để dễ trực quan hóa, đồng thời vẫn giữ được phần lớn thông tin.
-

Bước 6: Vẽ biểu đồ phân cụm theo PCA

```
for cluster in range(optimal_k):
```

```
    clustered = df[df["Cluster"] == cluster]
```

```
    plt.scatter(clustered["PCA1"], clustered["PCA2"], ...)
```

- Mỗi cụm sẽ được vẽ thành một nhóm điểm trên mặt phẳng 2D.
 - Mỗi màu tương ứng với một cụm.
-

Bước 7: Lưu kết quả

```
df.to_csv("clustered_results.csv", index=False)
```

- Lưu dữ liệu gốc cùng với nhãn cụm và tọa độ PCA vào file mới (clustered_results.csv).

Gợi ý cải tiến

- Thêm `print(df.groupby("Cluster").mean())` để hiểu trung bình các thông số của mỗi cụm.
- Dùng `seaborn` để có biểu đồ đẹp hơn (`sns.scatterplot`).
- Dùng `t-SNE` nếu `PCA` không cho kết quả trực quan tốt.

Bạn muốn mình minh họa sơ đồ phân cụm bằng dữ liệu mẫu hay bạn đã có file `results.csv` sẵn?

Bài tập 4:

1. Các thư viện sử dụng

Mã Python sử dụng các thư viện phổ biến sau:

- `requests`: Được sử dụng để gửi các yêu cầu HTTP GET tới các trang web và lấy dữ liệu HTML về.
- `BeautifulSoup` (từ `bs4`): Dùng để phân tích và trích xuất dữ liệu từ HTML. Nó giúp dễ dàng tìm kiếm các phần tử HTML (như bảng dữ liệu) và trích xuất nội dung.

- pandas: Đây là thư viện xử lý và phân tích dữ liệu mạnh mẽ, đặc biệt là với các dữ liệu dạng bảng (DataFrame). Nó giúp bạn dễ dàng thao tác với dữ liệu, lọc, sắp xếp, và lưu trữ vào các file (CSV, Excel, v.v.).
- openpyxl: Thư viện dùng để làm việc với file Excel (.xlsx). Trong mã này, nó được sử dụng để lưu kết quả vào file Excel.

2. Các hàm và chức năng trong mã

Mã chia thành ba hàm chính:

a. Hàm `get_fbref_data()`

- **Mục đích:** Tải và xử lý dữ liệu cầu thủ từ FBref, sau đó lọc các cầu thủ có thời gian thi đấu > 900 phút.
- **Chi tiết hoạt động:**
 - URL: Trang web chứa thống kê cầu thủ Premier League mùa 2024-2025.
 - `requests.get()`: Gửi yêu cầu HTTP để tải nội dung trang web.
 - BeautifulSoup: Phân tích cú pháp HTML của trang để tìm bảng chứa thống kê cầu thủ.
 - `pandas.read_html()`: Đọc bảng HTML và chuyển thành DataFrame.

- `df['Min']`: Chuyển đổi cột "Min" (phút thi đấu) thành kiểu dữ liệu số để dễ dàng lọc.
- Lọc cầu thủ: Dữ liệu được lọc để chỉ giữ lại những cầu thủ có thời gian thi đấu > 900 phút.
- Trả về DataFrame: Chỉ giữ các cột quan trọng như tên cầu thủ, CLB, số phút thi đấu, bàn thắng, kiến tạo, v.v.

b. Hàm `get_transfer_data()`

- Mục đích: Tải và xử lý dữ liệu chuyển nhượng từ FootballTransfers.
- Chi tiết hoạt động:
 - URL: Trang web chứa thông tin về các thương vụ chuyển nhượng cầu thủ.
 - `requests.get()`: Gửi yêu cầu HTTP để tải nội dung trang web.
 - BeautifulSoup: Phân tích cú pháp HTML của trang để tìm bảng chứa thông tin chuyển nhượng.
 - `pandas.read_html()`: Đọc bảng HTML và chuyển thành DataFrame.
 - Chọn các cột: Lọc các cột cần thiết như tên cầu thủ, CLB cũ, CLB mới và giá trị chuyển nhượng.
 - Trả về DataFrame: Dữ liệu chuyển nhượng đã được làm sạch.

c. Hàm `combine_data()`

- **Mục đích:** Kết hợp dữ liệu cầu thủ từ FBref và thông tin chuyển nhượng từ FootballTransfers.
- **Chi tiết hoạt động:**
 - Kết hợp hai DataFrame: Dữ liệu từ FBref và FootballTransfers được kết hợp thông qua tên cầu thủ bằng cách sử dụng `pd.merge()`.
 - Lưu vào Excel: Kết quả được lưu vào một file Excel sử dụng `to_excel()` từ thư viện pandas.

3. Quá trình và logic tổng thể

Bước 1: Lấy dữ liệu từ FBref

- Truy vấn dữ liệu từ bảng thống kê cầu thủ của Premier League (mùa giải 2024-2025).
- Chuyển đổi dữ liệu: Đảm bảo rằng các giá trị thời gian thi đấu là kiểu số và có thể lọc theo điều kiện > 900 phút.

Bước 2: Lấy dữ liệu chuyển nhượng từ FootballTransfers

- Truy xuất dữ liệu chuyển nhượng từ FootballTransfers để lấy thông tin về giá trị chuyển nhượng, CLB cũ và CLB mới của các cầu thủ.

Bước 3: Lọc cầu thủ có thời gian thi đấu > 900 phút

- Lọc dữ liệu từ bảng FBref để chỉ giữ lại cầu thủ có số phút thi đấu trên 900.

Bước 4: Kết hợp dữ liệu

- Kết hợp thông tin cầu thủ với dữ liệu chuyển nhượng, đảm bảo chỉ giữ lại những cầu thủ có thời gian thi đấu > 900 phút và có thông tin chuyển nhượng.

Bước 5: Lưu kết quả vào Excel

- Lưu file Excel chứa thông tin cầu thủ và dữ liệu chuyển nhượng để tiện theo dõi và phân tích.

4. Kết quả

Player	From Club	To Club	Transfer Fee	Minutes
Julián Álvarez	Man City	Atlético Madrid	€75M	1250
Dominic Solanke	Bournemouth	Tottenham	€64.3M	2020
Leny Yoro	Lille	Man United	€62M	1350
Pedro Neto	Wolves	Chelsea	€60M	910
João Neves	Benfica	PSG	€59.92M	1500
Amadou Onana	Everton	Aston Villa	£50M	1230
Joshua Zirkzee	Bologna	Man United	€42.5M	980
Evanilson	Porto	Bournemouth	£40.2M	1120