HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

o0o

BÁO CÁO BÀI TẬP LỚN 1

NGÔN NGỮ LẬP TRÌNH PYTHON

Giảng viên hướng dẫn:	**Kim Ngọc Bách**
Sinh viên:	**Vũ Thị Thu Duyên**
Mã sinh viên:	**B23DCCE027**
Lớp:	**D23CQCEO6-B**
Niên khóa:	**2023 - 2028**
Hệ đào tạo:	**Đại học chính quy**

Hà Nội, 2025

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm: **(Bằng chữ:)**

Hà Nội, ngày tháng năm 20...

Giảng viên

Mục lục

Danh sách hình vẽ

Chương 1

Thu thập dữ liệu thống kê cầu thủ ngoại hạng anh mùa 2024-2025 từ fbref.com

1.1 Giới thiệu

Bài tập này yêu cầu xây dựng một chương trình Python để tự động thu thập dữ liệu thống kê chi tiết của các cầu thủ bóng đá tại giải Ngoại hạng Anh (English Premier League - EPL) mùa giải 2024-2025 từ trang web `fbref.com`. Chương trình sẽ tạo ra hai tệp kết quả:

> `results.csv`: Chứa dữ liệu của các cầu thủ có tổng số phút thi đấu trong cả mùa giải (cho một hoặc nhiều CLB) lớn hơn 90 phút, theo yêu cầu cụ thể của bài tập này.
>
> `results1.csv`: Chứa dữ liệu của các cầu thủ thi đấu hơn 90 phút cho một câu lạc bộ cụ thể trong mùa giải. Tệp này giữ lại các bản ghi riêng lẻ theo từng đội và sẽ là dữ liệu đầu vào quan trọng cho các bài tập phân tích và tổng hợp thống kê tiếp theo, nơi cần dữ liệu chi tiết theo từng lần khoác áo CLB của cầu thủ.

Dưới đây sẽ là phần mô tả quá trình thực hiện, bao gồm lựa chọn thư viện, quy trình cào dữ liệu và kết quả thu được.

1.2 Lựa chọn thư viện

Để thực hiện nhiệm vụ cào dữ liệu từ một trang web động và xử lý dữ liệu hiệu quả, các thư viện Python sau đã được lựa chọn:

> **Selenium**: Thư viện này được chọn làm công cụ chính để tương tác với trang web FBRef vì trang này sử dụng nhiều JavaScript để tải và hiển thị nội dung, đặc biệt là các bảng dữ liệu thống kê phức tạp. Không giống như thư viện `requests` (vốn chỉ lấy mã nguồn HTML tĩnh ban đầu do máy chủ gửi về và không thể thực thi JavaScript), `selenium` hoạt động bằng cách điều khiển một trình duyệt web thực sự (như Chrome hoặc Firefox) một cách tự động. Điều này cho phép script chờ đợi các yếu tố động (như bảng dữ liệu) được JavaScript tải xong và hiển thị đầy đủ, thực thi bất kỳ mã JavaScript nào trên trang, và truy cập vào cấu trúc HTML cuối cùng (DOM) đã được render hoàn chỉnh. Nếu chỉ dùng `requests`, có nguy cơ rất

cao sẽ bỏ lỡ các bảng dữ liệu quan trọng hoặc chỉ nhận được dữ liệu không đầy đủ, vì chúng được tạo ra hoặc điền dữ liệu bởi JavaScript sau khi trang tải lần đầu.

BeautifulSoup4 (bs4): Sau khi `selenium` tải xong trang và lấy được mã nguồn HTML, `BeautifulSoup` được sử dụng để phân tích cú pháp (parse) HTML đó. Nó cung cấp các phương thức tiện lợi để điều hướng HTML, tìm kiếm các thẻ (tags), thuộc tính (attributes) cụ thể (như `id`, `class`, `data-stat`), và trích xuất nội dung văn bản một cách dễ dàng và hiệu quả.

Pandas: Đây là thư viện mạnh mẽ cho thao tác và phân tích dữ liệu

Json: Tệp `config.json` chứa các cấu hình quan trọng như URL cơ sở, tên các chỉ số cần lấy, ID bảng dữ liệu tương ứng, thuộc tính `data-stat`, ngưỡng số phút tối thiểu, v.v. Thư viện `json` được dùng để đọc và phân tích cú pháp tệp cấu hình này, giúp mã nguồn chính trở nên gọn gàng, dễ đọc và dễ bảo trì hơn. Việc tách cấu hình ra giúp dễ dàng thay đổi các tham số hoặc cập nhật cấu trúc trang web mà không cần sửa đổi nhiều trong mã lệnh Python.

Re (Regular Expressions): Được sử dụng để trích xuất ID duy nhất của cầu thủ từ các đường dẫn URL trong mã HTML (`/players/<player_id>/`). ID này rất quan trọng để định danh và tổng hợp dữ liệu cho cùng một cầu thủ khi họ xuất hiện ở nhiều bảng thống kê khác nhau hoặc thi đấu cho nhiều CLB trong mùa giải.

Time: Được sử dụng để tạo độ trễ nhỏ (`time.sleep`) giữa các yêu cầu truy cập trang web, nhằm tránh việc gửi quá nhiều yêu cầu liên tục đến máy chủ FBRef, giảm nguy cơ bị chặn IP và thể hiện hành vi duyệt web "lịch sự".

Collections.defaultdict: Cực kỳ hữu ích khi cần tổng hợp dữ liệu. Trong trường hợp một cầu thủ thi đấu cho nhiều câu lạc bộ trong cùng mùa giải (ví dụ: chuyển nhượng giữa mùa), họ sẽ xuất hiện nhiều lần trong bảng dữ liệu. `defaultdict` được dùng để tạo một cấu trúc dữ liệu cho phép cộng dồn số phút thi đấu (`Min`) của cầu thủ đó từ các dòng khác nhau, trước khi áp dụng bộ lọc số phút tối thiểu.

1.3 Quá trình cào dữ liệu (Web Scraping)

Ý tưởng chính của script này là tự động hóa quá trình thu thập dữ liệu từ một nguồn web động (fbref.com) bằng cách kết hợp Selenium để mô phỏng hành vi của người dùng và BeautifulSoup để phân tích cấu trúc HTML. Dữ liệu cần lấy được xác định dựa trên cấu hình bên ngoài (`config.json`), cho phép linh hoạt lựa chọn các thống kê cần thiết từ nhiều bảng khác nhau trên trang. Điểm mấu chốt trong việc xử lý dữ liệu là chương trình ban đầu định danh mỗi bản ghi thống kê riêng lẻ theo cặp (ID cầu thủ duy nhất hoặc tên, tên đội). Việc này là cần thiết để phân biệt chính xác dữ liệu của cùng một cầu thủ khi họ thi đấu cho các câu lạc bộ khác nhau trong cùng một mùa giải (ví dụ: sau khi chuyển nhượng), bởi trên fbref, cầu thủ có thể xuất hiện ở nhiều dòng khác nhau nếu họ đổi đội. Sau đó, dữ liệu này được xử lý theo hai cách riêng biệt để tạo ra hai tệp CSV:

- **results1.csv**: Lọc trực tiếp từ dữ liệu chi tiết, giữ lại những bản ghi mà cầu thủ thi đấu > 90 phút cho CLB đó.
- **results.csv**: Tổng hợp tổng số phút thi đấu của từng cầu thủ trên toàn mùa giải, sau đó lọc những cầu thủ có tổng số phút > 90 và xuất ra tất cả các bản ghi liên

quan đến cầu thủ đó (kể cả những bản ghi dưới 90 phút cho một CLB cụ thể, miễn là tổng số phút đạt yêu cầu).

Quy trình cào dữ liệu được thực hiện theo các bước sau:

Bước 1: Tải cấu hình: Chương trình bắt đầu bằng việc đọc tệp `config.json` để lấy các thông tin cần thiết như URL gốc của mùa giải EPL trên fbref, danh sách các chỉ số thống kê cần thu thập, ID của các bảng HTML chứa dữ liệu, thuộc tính `data-stat` tương ứng với từng chỉ số, ngưỡng số phút thi đấu tối thiểu (`min_minutes`), tên tệp CSV đầu ra, và thứ tự các cột trong tệp kết quả.

Bước 2: Khởi tạo selenium: Một trình duyệt web tự động (WebDriver) được khởi tạo bằng `selenium`.

Bước 3: Truy cập và tải trang:

- Chương trình duyệt đến URL của bảng thống kê chính (`stats`) cho mùa giải EPL 2024-2025.
- Sử dụng `WebDriverWait` để chờ cho đến khi bảng dữ liệu chính (`stats_standard`) xuất hiện và có thể tương tác, đảm bảo rằng mọi dữ liệu tải bằng JavaScript đã được hiển thị.

Bước 4: Phân tích HTML ban đầu: Mã nguồn HTML của trang đã tải hoàn chỉnh được lấy từ `selenium` và chuyển cho `BeautifulSoup` để phân tích cú pháp.

Bước 5: Thu thập dữ liệu từ nhiều bảng:

- Script được thiết kế để lấy dữ liệu từ nhiều loại bảng thống kê khác nhau trên fbref (Standard, Shooting, Passing, Defensive Actions, etc.), như được định nghĩa trong `STATS_CONFIG` của tệp `config.json`.
- Đối với mỗi loại thống kê, script xây dựng URL tương ứng (ví dụ: `.../stats/`, `.../shooting/`, `.../passing/`).
- Script truy cập từng URL này bằng `selenium`, đợi bảng dữ liệu tương ứng tải xong, lấy HTML và dùng `BeautifulSoup` để phân tích.
- Lặp qua từng dòng (`<tr>`) trong phần thân (`<tbody>`) của bảng dữ liệu.

Bước 6: Trích xuất dữ liệu cầu thủ: Với mỗi dòng (đại diện cho một cầu thủ trong bảng đó), script thực hiện:

- Tìm ô chứa tên cầu thủ và trích xuất tên (`Player`) và ID duy nhất của cầu thủ (`player_id`) từ liên kết hồ sơ cầu thủ bằng biểu thức chính quy (`re`).
- Sử dụng hàm `safe_get_text`, `get_nation_code` và cấu hình `data-stat` từ `config.json` để trích xuất giá trị của từng chỉ số thống kê được yêu cầu cho cầu thủ đó.
- Hàm này đảm bảo trả về "N/a"nếu một ô dữ liệu không tồn tại hoặc trống.

Bước 7: Lưu trữ tạm thời: Dữ liệu của mỗi cầu thủ từ mỗi bảng được lưu vào một dictionary, sau đó được thêm vào một danh sách tổng hợp (`scraped_data`).

Bước 8: Tổng hợp và Lọc Dữ liệu (2 luồng):

- Luồng 1: Tạo dữ liệu cho `results1.csv`

- Lặp qua `scraped_data`.
- Với mỗi bản ghi (ứng với một cầu thủ tại một đội), kiểm tra giá trị `Playing Time: minutes ('Min')`.
- Nếu số phút > `MIN_MINUTES`, định dạng lại các giá trị (dùng `format_value`) và thêm dictionary của bản ghi này vào danh sách `single_team_over_90_list`.

- Luồng 2: Tạo dữ liệu cho `results.csv`
  - Sử dụng `collections.defaultdict` (`player_minutes_aggregate`) để nhóm các bản ghi theo `player_id` (hoặc `player_name` nếu ID không có).
  - Trong quá trình nhóm, cộng dồn số phút thi đấu (`Min`) để tính `total_minutes` cho mỗi cầu thủ.
  - Lọc `player_minutes_aggregate`, chỉ giữ lại những cầu thủ có `total_minutes` > `MIN_MINUTES`.
  - Với mỗi cầu thủ đủ điều kiện, lặp qua *tất cả* các bản ghi gốc (entries) của cầu thủ đó (từ các đội khác nhau nếu có).
  - Định dạng lại các giá trị cho từng bản ghi và thêm vào danh sách `final_player_list_of_dicts`.

Bước 9: Định dạng và Sắp xếp:

- Cả hai danh sách (`single_team_over_90_list` và `final_player_list_of_dicts`) được sắp xếp theo tên (first name) của cầu thủ theo thứ tự bảng chữ cái.
- Đảm bảo tất cả các cột theo `COLUMN_ORDER` đều tồn tại, điền "N/a"nếu thiếu (đã thực hiện trong bước định dạng ở Bước 6).

Bước 10: Xuất ra CSV:

- Chuyển danh sách `final_player_list_of_dicts` thành pandas DataFrame và lưu vào tệp `results.csv` (được đặt tên trong `config.json`).
- Chuyển danh sách `single_team_over_90_list` thành pandas DataFrame và lưu vào tệp `results1.csv`.
- Cả hai tệp CSV đều không bao gồm chỉ số dòng (index) của DataFrame và sử dụng encoding `utf-8-sig` để hiển thị ký tự đặc biệt đúng cách.

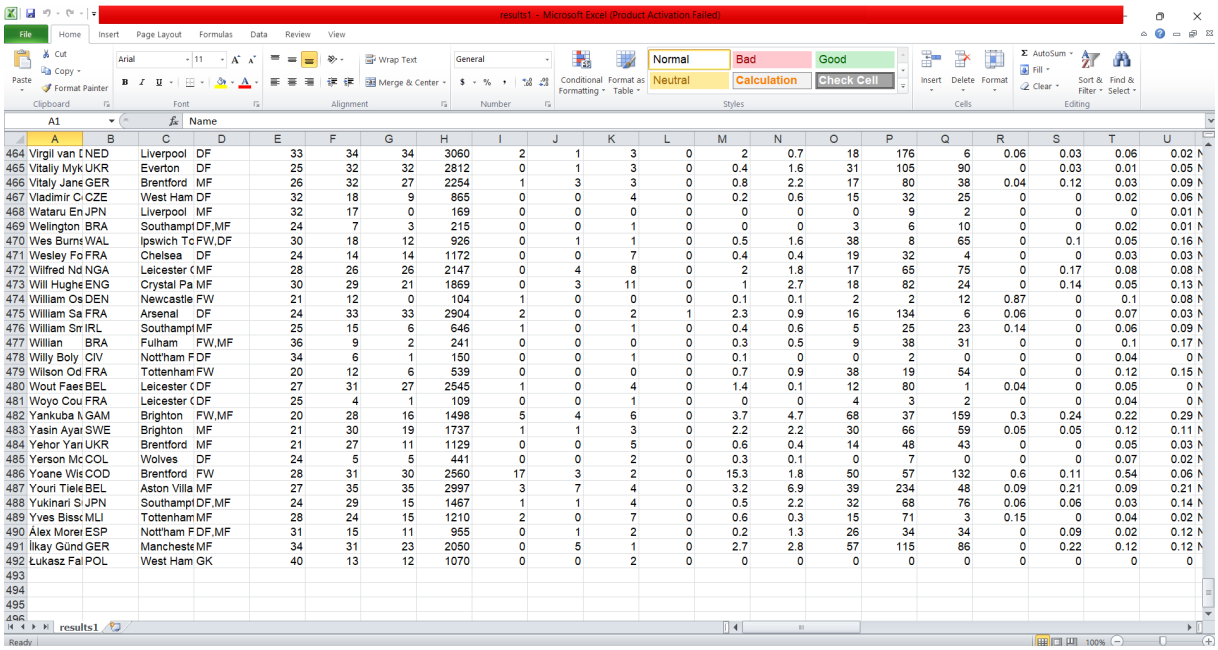
1.4 Kết quả

Chương trình thực thi và tạo ra hai tệp CSV:

`results1.csv`: Đã thu thập dữ liệu thống kê cho 491 cầu thủ, mỗi người đạt trên 90 phút thi đấu cho một câu lạc bộ ở giải Ngoại hạng Anh (EPL) mùa 2024-2025.

`results.csv`: Đã thu thập dữ liệu thống kê cho 496 cầu thủ, mỗi người đạt trên 90 phút thi đấu trong cả mùa 2024-2025.

Việc tạo ra `results1.csv` đảm bảo dữ liệu gốc (đã lọc sơ bộ theo tiêu chí phút/đội) có sẵn cho các bước xử lý tiếp theo, trong khi `results.csv` giải quyết yêu cầu trực tiếp của bài 1 này.



Hình 1.1: Hình ảnh file results.csv

Chương 2

Phân tích và trực quan hóa dữ liệu thống kê

2.1 Phân tích top 3 cầu thủ cao nhất và thấp nhất theo từng chỉ số thống kê

2.1.1 Khái quát về yêu cầu của bài toán

Bài toán yêu cầu xử lý một tệp dữ liệu (`results1.csv` từ bài tập trước) chứa thông tin thống kê của các cầu thủ bóng đá. Mục tiêu chính là xác định và liệt kê 3 cầu thủ có điểm số cao nhất (Top 3) và 3 cầu thủ có điểm số thấp nhất (Bottom 3) cho mỗi chỉ số thống kê có trong tệp dữ liệu. Kết quả phân tích này được lưu vào một tệp văn bản có tên là `top_3_formatted.txt`.

2.1.2 Khái quát logic code chính (`top_3_statistical_analysis.py`)

Script sử dụng thư viện `pandas` để đọc dữ liệu từ tệp CSV. Sau đó, lặp qua từng cột thống kê (loại trừ các cột thông tin cơ bản như tên, quốc tịch, đội, vị trí). Đối với mỗi cột thống kê, code cố gắng chuyển đổi dữ liệu sang dạng số, loại bỏ các giá trị không hợp lệ (`N/a` hoặc không thể chuyển đổi). Nếu cột có thể được coi là số và có dữ liệu hợp lệ, code sẽ sắp xếp dữ liệu để tìm ra 3 giá trị cao nhất và 3 giá trị thấp nhất cùng với tên cầu thủ và đội tương ứng. Cuối cùng, kết quả được định dạng và ghi vào tệp `top_3_formatted.txt`.

2.1.3 Xử lý 'N/a'

Trong quá trình xử lý dữ liệu, các giá trị `'N/a'` được xem là dữ liệu thiếu và được chuyển thành `NaN` thông qua tham số `na_values=['N/a']` khi đọc file CSV bằng Pandas. Ngoài ra, các giá trị không thể chuyển đổi sang dạng số cũng được ép về `NaN` bằng `pd.to_numeric(errors='coerce')`. Thay vì thay thế các giá trị thiếu bằng một con số cụ thể (như 0), toàn bộ các dòng chứa `NaN` ở cột cần đánh giá sẽ được loại bỏ trước khi thực hiện việc xếp hạng. Việc này được thực hiện dựa trên các nguyên tắc sau:

- **Tính đúng đắn của dữ liệu:** `NaN` thể hiện rằng dữ liệu bị thiếu hoặc không áp dụng, ví dụ như thông số dành riêng cho một vị trí cụ thể không phù hợp với cầu thủ ở vị trí khác. Việc thay thế bằng 0 sẽ làm sai lệch bản chất này.

- **Tránh gây hiểu nhầm:** Gán giá trị 0 cho dữ liệu thiếu có thể dẫn đến diễn giải sai – ví dụ, hiểu nhầm rằng cầu thủ không ghi bàn, trong khi thực tế dữ liệu chưa được ghi nhận.
- **Đảm bảo công bằng khi xếp hạng:** Việc chỉ xét các cầu thủ có dữ liệu hợp lệ giúp kết quả Top/Bottom 3 phản ánh đúng hiệu suất thực tế, tránh việc đưa các cá nhân thiếu dữ liệu vào so sánh một cách không chính xác.

Như vậy, việc giữ nguyên `NaN` và loại bỏ các giá trị thiếu khỏi quá trình xếp hạng là phương pháp hợp lý, đảm bảo tính khách quan và chính xác trong phân tích dữ liệu cầu thủ.

2.1.4 Các bước thực thi

Khởi tạo và chuẩn bị dữ liệu:

Script nhập các thư viện cần thiết (`pandas`, `os`, `numpy`, `io`). Đọc dữ liệu từ file `results.csv` vào DataFrame Pandas, cấu hình để xử lý `'N/a'` thành `NaN`. Xác định danh sách các cột thống kê (`stats_columns`) cần phân tích bằng cách loại trừ các cột định danh ('Name', 'Team', 'Position', 'Nation').

Xử lý từng cột và ghi file output:

Mở file `top_3_formatted.txt` để ghi kết quả. Script lặp qua từng cột `col` trong `stats_columns`:

- Ghi Tiêu đề Cột: Ghi tên cột hiện tại vào file.
- Chuẩn hóa & Lọc Dữ liệu Số: Sử dụng `pd.to_numeric(errors='coerce')` để ép kiểu cột `col` sang dạng số, chuyển lỗi thành `NaN`.
- Tạo DataFrame tạm `df_for_sort` và loại bỏ các hàng có giá trị `NaN` (`dropna()`) trong cột này.
- Bước này đảm bảo chỉ dữ liệu số, hợp lệ được sử dụng để xếp hạng.
- Xếp hạng và Ghi Kết quả:
  - Nếu `df_for_sort` chứa dữ liệu số hợp lệ: Dùng `sort_values()` để sắp xếp DataFrame này theo cột `col` giảm dần (tìm Top 3) và tăng dần (tìm Bottom 3), sau đó dùng `head(3)`.
  - Kết quả (Tên, Đội, Chỉ số) và thông tin kiểu dữ liệu được ghi vào file.
- Ghi Phân cách: Thêm dòng `===...` để phân tách kết quả giữa các cột.

Hoàn tất: Sau khi xử lý hết các cột, file `top_3_formatted.txt` được lưu lại, chứa toàn bộ kết quả phân tích.

2.1.5 Kết quả (`top_3_formatted.txt`)

Tệp `top_3_formatted.txt` chứa kết quả phân tích cho từng chỉ số thống kê. Mỗi chỉ số được trình bày rõ ràng với tiêu đề, tiếp theo là danh sách Top 3 và Bottom 3 cầu thủ.

```
15  ================================================

16

17  --- Playing Time: matches played ---

18

19  Top 3 Players:
20                  Name      Team  Playing Time: matches played
21  Ryan Gravenberch Liverpool                              34
22      Raúl Jiménez    Fulham                              34
23    Mohamed Salah Liverpool                               34

24

25  Bottom 3 Players:
26                  Name           Team  Playing Time: matches played
27     Reiss Nelson        Arsenal                               1
28  Joachim Andersen Crystal Palace                              1
29    Carlos Alcaraz    Southampton                              1

30

31  ================================================
```

Hình 2.1: Ví dụ kết quả từ tệp top_3_formatted.txt.

2.2 Tính toán thống kê mô tả (median, mean, std dev) cho dữ liệu cầu thủ

2.2.1 Giới thiệu

Bài tập này yêu cầu viết một chương trình Python để thực hiện các phép tính thống kê mô tả trên dữ liệu cầu thủ đã thu thập ở bài tập trước (`results1.csv`). Cụ thể, chương trình cần tính toán:

Giá trị trung vị (median) cho mỗi chỉ số thống kê trên toàn bộ tập dữ liệu.

Giá trị trung bình (mean) và độ lệch chuẩn (standard deviation) cho mỗi chỉ số thống kê, tính toán trên toàn bộ tập dữ liệu và tính riêng cho từng đội (team).

Kết quả được lưu vào tệp `results2.csv` với một định dạng cụ thể, trong đó các hàng đại diện cho toàn bộ ('all') hoặc một đội cụ thể, và các cột đại diện cho các phép tính thống kê (Median, Mean, Std) áp dụng cho từng chỉ số.

2.2.2 Lựa chọn thư viện

Thư viện `pandas`: Là thư viện chủ đạo, không thể thiếu cho tác vụ này.

Thư viện `os`: Được sử dụng để quản lý đường dẫn tệp một cách linh hoạt và độc lập với hệ điều hành. `os.path.join` và `os.path.dirname` giúp xác định chính xác vị trí của tệp đầu vào (`results1.csv`) và tệp đầu ra (`results2.csv`) mà không cần mã hóa cứng đường dẫn.

2.2.3 Logic và quy trình thực hiện

Chương trình `calculating_statistics.py` thực hiện các bước sau:

Bước 1: Chuẩn bị và đọc dữ liệu: Sử dụng `os` để xây dựng đường dẫn đến tệp input `results1.csv` và tệp output `results2.csv`. Đọc `results.csv` vào pandas DataFrame (`df`). Một danh sách `na_values_list` gồm nhiều biến thể của giá trị thiếu ('N/a', 'n/a', '', ...) được cung cấp cho tham số `na_values` của `pd.read_csv` để đảm bảo nhận diện đúng các giá trị này và chuyển đổi chúng thành `NaN` (Not a Number) của pandas, thuận lợi cho các phép tính sau này.

Bước 2: Xác định cột thống kê: Định nghĩa danh sách `exclude_cols` chứa các cột không phải là số liệu thống kê cần phân tích (ví dụ: 'Name', 'Nation', 'Team', 'Position', 'Age'). Tạo danh sách `stats_columns` bằng cách lọc các cột trong DataFrame, chỉ giữ lại những cột không nằm trong `exclude_cols`.

Bước 3: Tính toán thống kê tổng thể ('all'): Chọn các cột trong `stats_columns` từ DataFrame `df`. Sử dụng phương thức `.agg(['median', 'mean', 'std'])` để tính đồng thời trung vị, trung bình và độ lệch chuẩn cho từng cột trong `stats_columns`. Kết quả trả về có dạng các chỉ số thống kê là hàng, các cột gốc là cột. Sử dụng `.T` (transpose) để chuyển vị, đưa tên các chỉ số gốc (`stats_columns`) thành chỉ số hàng, và 'median', 'mean', 'std' thành tên cột, lưu vào `overall_stats`. Điều này giúp truy cập dễ dàng hơn ở bước sau.

Bước 4: Tính toán thống kê theo nhóm ('Team'): Sử dụng `df.groupby('Team')` để nhóm DataFrame theo giá trị trong cột 'Team'. Trên đối tượng GroupBy này, chọn các cột `stats_columns`. Áp dụng `.agg(['median', 'mean', 'std'])` để tính toán các chỉ số thống kê cho từng đội. Kết quả (`team_stats`) sẽ có cấu trúc đa chỉ số (MultiIndex) ở cả hàng (Team) và cột (Statistic, Metric).

Bước 5: Tái cấu trúc dữ liệu cho đầu ra:

- Tạo hàng 'all': Khởi tạo một DataFrame rỗng `all_row` với chỉ số là 'all'. Lặp qua từng cột thống kê (`col` in `stats_columns`), tạo ra các cột mới trong `all_row` với tên theo định dạng `f'Median of {col}'`, `f'Mean of {col}'`, `f'Std of {col}'` và gán giá trị tương ứng lấy từ `overall_stats` đã tính ở Bước 3.
- Tạo các hàng 'Team': Khởi tạo một DataFrame `team_results` với chỉ số là tên các đội (lấy từ `team_stats.index`). Lặp qua từng cột thống kê (`col` in `stats_columns`), tạo ra các cột mới trong `team_results` với tên định dạng tương tự như trên. Giá trị được lấy từ `team_stats` bằng cách truy cập qua chỉ số đa cấp, ví dụ: `team_stats[(col, 'median')]` để lấy cột trung vị của chỉ số `col`.
- Kết hợp Kết quả: Sử dụng `pd.concat([all_row, team_results])` để nối DataFrame chứa hàng 'all' và DataFrame chứa các hàng của từng đội lại với nhau theo chiều dọc, tạo thành DataFrame cuối cùng `final_results` có cấu trúc đúng yêu cầu.

Bước 6: Lưu Kết quả: Xuất DataFrame `final_results` ra tệp `results2.csv` bằng phương thức `.to_csv()`. Tham số `index=True` được sử dụng (mặc định) để lưu chỉ số của DataFrame (chính là 'all' hoặc tên đội) vào cột đầu tiên của tệp CSV.

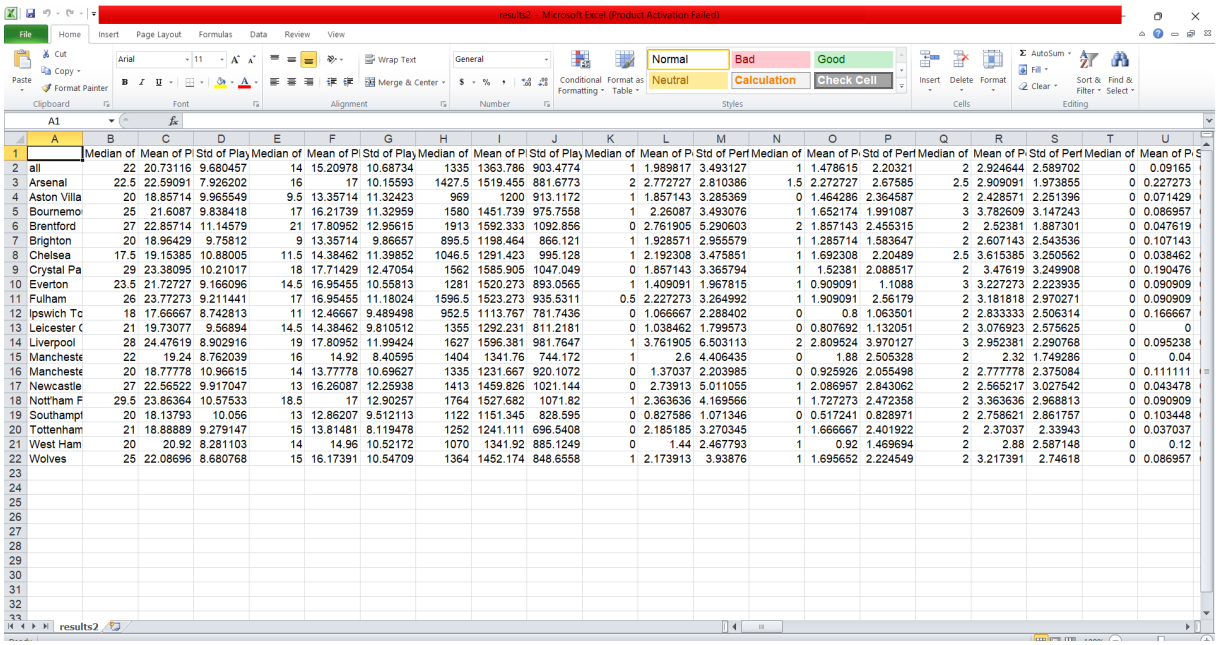
2.2.4 Kết quả

Chương trình đã chạy thành công và tạo ra tệp `results2.csv`. Tệp `results2.csv` chứa các kết quả thống kê tổng hợp theo định dạng yêu cầu:

- Hàng: Hàng đầu tiên có chỉ số là 'all', đại diện cho thống kê trên toàn bộ cầu thủ. Các hàng tiếp theo có chỉ số là tên của từng đội, đại diện cho thống kê tính riêng cho cầu thủ của đội đó.

- Cột: Các cột được đặt tên theo mẫu "Metric of Statistic", ví dụ: "Median of Performance: goals", "Mean of Performance: goals", "Std of Performance: goals", "Median of Performance: assists", v.v., bao gồm tất cả các chỉ số trong `stats_columns`.

- Giá trị: Các ô chứa giá trị trung vị, trung bình hoặc độ lệch chuẩn tương ứng đã được tính toán. Các giá trị NaN (nếu có, ví dụ: độ lệch chuẩn của nhóm chỉ có 1 cầu thủ) sẽ được biểu diễn dưới dạng ô trống trong tệp CSV.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		Median of	Mean of Pl	Std of Play	Median of	Mean of Pl	Std of Play	Median of	Mean of Pl	Std of Play	Median of	Mean of Pl	Std of Perf	Median of	Mean of Pl	Std of Perf	Median of	Mean of Pl	Std of Perf	Median of	Mean of Pl
2	all	22	20.73116	9.680457	14	15.20978	10.68734	1335	1363.786	903.4774	1	1.989817	3.493127	1	1.478615	2.20321	2	2.924644	2.589702	0	0.09165
3	Arsenal	22.5	22.59091	7.926202	16	17	10.15593	1427.5	1519.455	881.6773	2	2.772727	2.810386	1.5	2.272727	2.67585	2.5	2.909091	1.973855	0	0.227273
4	Aston Villa	20	18.85714	9.965549	9.5	13.35714	11.32423	969	1200	913.1172	1	1.857143	3.285369	0	1.464286	2.364587	2	2.428571	2.251396	0	0.071429
5	Bournemo	25	21.6087	9.838418	17	16.21739	11.32959	1580	1451.739	975.7558	1	2.26087	3.493076	1	1.652174	1.991087	3	3.782609	3.147243	0	0.086957
6	Brentford	27	22.85714	11.14579	21	17.80952	12.95615	1913	1592.333	1092.856	0	2.761905	5.290603	2	1.857143	2.455315	2	2.52381	1.887301	0	0.047619
7	Brighton	20	18.96429	9.75812	9	13.35714	9.86657	895.5	1198.464	866.121	1	1.928571	2.955579	1	1.285714	1.583647	2	2.607143	2.543536	0	0.107143
8	Chelsea	17.5	19.15385	10.88005	11.5	14.38462	11.39852	1046.5	1291.423	995.128	1	2.192308	3.475851	1	1.692308	2.20489	2.5	3.615385	3.250562	0	0.038462
9	Crystal Pa	29	23.38095	10.21017	18	17.71429	12.47054	1562	1585.905	1047.049	0	1.857143	3.365794	1	1.52381	2.088517	2	3.47619	3.249908	0	0.190476
10	Everton	23.5	21.72727	9.166096	14.5	16.95455	10.55813	1281	1520.273	893.0565	1	1.409091	1.967815	1	0.909091	1.1088	3	3.227273	2.223935	0	0.090909
11	Fulham	26	23.77273	9.211441	17	16.95455	11.18024	1596.5	1523.273	935.5311	0.5	2.227273	3.264992	1	1.909091	2.56179	2	3.181818	2.970271	0	0.090909
12	Ipswich Tc	18	17.66667	8.742813	11	12.46667	9.489498	952.5	1113.767	781.7436	0	1.066667	2.288402	0	0.8	1.063501	2	2.833333	2.506314	0	0.166667
13	Leicester C	21	19.73077	9.56894	14.5	14.38462	9.810512	1355	1292.231	811.2181	0	1.038462	1.799573	0	0.807692	1.132051	2	3.076923	2.575625	0	0
14	Liverpool	28	24.47619	8.902916	19	17.80952	11.99424	1627	1596.381	981.7647	1	3.761905	6.503113	2	2.809524	3.970127	3	2.952381	2.290768	0	0.095238
15	Mancheste	22	19.24	8.762039	16	14.92	8.40595	1404	1341.76	744.172	1	2.6	4.406435	0	1.88	2.505328	2	2.32	1.749286	0	0.04
16	Mancheste	20	18.77778	10.96615	14	13.77778	10.69627	1335	1231.667	920.1072	0	1.37037	2.203985	0	0.925926	2.055498	2	2.777778	2.375084	0	0.111111
17	Newcastle	27	22.56522	9.917047	13	16.26087	12.25938	1413	1459.826	1021.144	0	2.73913	5.011055	1	2.086957	2.843062	2	2.565217	3.027542	0	0.043478
18	Nott'ham F	29.5	23.86364	10.57533	18.5	17	12.90257	1764	1527.682	1071.82	1	2.363636	4.169566	1	1.727273	2.472358	2	3.363636	2.968813	0	0.090909
19	Southampt	20	18.13793	10.056	13	12.86207	9.512113	1122	1151.345	828.595	0	0.827586	1.071346	0	0.517241	0.828971	2	2.758621	2.861757	0	0.103448
20	Tottenham	21	18.88889	9.279147	15	13.81481	8.119478	1252	1241.111	696.5408	0	2.185185	3.270345	1	1.666667	2.401922	2	2.37037	2.33943	0	0.037037
21	West Ham	20	20.92	8.281103	14	14.96	10.52172	1070	1341.92	885.1249	0	1.44	2.467793	1	0.92	1.469694	2	2.88	2.587148	0	0.12
22	Wolves	25	22.08696	8.680768	15	16.17391	10.54709	1364	1452.174	848.6558	1	2.173913	3.93876	1	1.695652	2.224549	2	3.217391	2.74618	0	0.086957

Hình 2.2: Ví dụ dữ liệu từ tệp results2.csv.

2.3 Trực quan hóa phân phối thống kê cầu thủ bằng biểu đồ histogram

2.3.1 Mục tiêu

Mục tiêu chính của script `plot_stat_histograms.py` là trực quan hóa và phân tích sự phân phối của một tập hợp các chỉ số thống kê quan trọng của cầu thủ bóng đá. Cụ thể, script nhằm:

- Tạo biểu đồ histogram cho từng chỉ số thống kê được chọn (`stats_to_plot`) để thể hiện sự phân phối của chỉ số đó trên *toàn bộ giải đấu* (tất cả cầu thủ).

- Tạo biểu đồ histogram riêng cho từng *đội bóng*, thể hiện sự phân phối của từng chỉ số thống kê trong nội bộ đội đó.

- Lưu các biểu đồ này dưới dạng file ảnh (PNG) để dễ dàng xem xét, chia sẻ và phân tích sau này.

- Cung cấp cái nhìn trực quan về hình dạng phân phối (ví dụ: đối xứng, lệch trái, lệch phải), xu hướng trung tâm, và độ phân tán của các chỉ số thống kê, cả ở cấp độ giải đấu và cấp độ đội bóng.

Các chỉ số thống kê cụ thể được phân tích trong script này bao gồm: `'Performance: goals'`, `'Performance: assists'`, `'Standard: shoots on target percentage (SoT%)'`, `'Blocks: Int'`, `'Performance: Recov'`, `'Challenges: Att'`.

2.3.2 Lựa chọn Thư viện

Script sử dụng một số thư viện Python phổ biến cho xử lý dữ liệu và trực quan hóa:

- `pandas`: Thư viện cốt lõi để đọc dữ liệu từ file CSV (`results1.csv`), lưu trữ dữ liệu dưới dạng DataFrame, và thực hiện các thao tác xử lý, lọc, và lựa chọn dữ liệu cần thiết cho việc vẽ biểu đồ. Nó cũng xử lý các giá trị thiếu (NaN).
- `matplotlib.pyplot`: Thư viện nền tảng cho việc tạo biểu đồ trong Python. Nó được sử dụng để tạo khung hình (figure), các trục (axes), đặt tiêu đề, nhãn, và quản lý layout của biểu đồ, đặc biệt khi vẽ nhiều biểu đồ con (subplots) cho các đội.
- `seaborn`: Thư viện xây dựng trên `matplotlib`, cung cấp giao diện cấp cao hơn để vẽ các biểu đồ thống kê hấp dẫn và giàu thông tin. Cụ thể, `seaborn.histplot` được dùng để vẽ biểu đồ histogram, bao gồm cả đường ước lượng mật độ xác suất giúp làm mịn và hình dung rõ hơn hình dạng phân phối (KDE - Kernel Density Estimate). `seaborn.set_theme` dùng để thiết lập phong cách thẩm mỹ chung cho các biểu đồ.
- `numpy`: Thư viện tính toán khoa học, được sử dụng trong hàm `calculate_fd_bins` để thực hiện các phép toán cần thiết (như căn bậc ba `np.cbrt`, logarit cơ số 2 `np.log2`) cho việc tính toán số lượng bins tối ưu theo quy tắc Freedman-Diaconis và Sturges.
- `os`: Thư viện cung cấp các hàm tương tác với hệ điều hành, chủ yếu được dùng để quản lý đường dẫn file (lấy đường dẫn thư mục hiện tại, nối đường dẫn), tạo thư mục lưu trữ (`stat_histograms`), và đảm bảo đường dẫn hợp lệ.
- `math`: Thư viện toán học cơ bản, được sử dụng cho các hàm như `math.ceil` để làm tròn lên khi tính toán số lượng hàng/cột cho subplot và số lượng bins.

2.3.3 Các bước thực hiện

Script thực hiện các bước chính sau:

Thiết lập cấu hình

Định nghĩa các hằng số như thư mục đầu ra (`OUTPUT_DIR`), tên cột chứa thông tin đội (`TEAM_COL`), số lượng đội tối đa trên mỗi biểu đồ (`TEAMS_PER_PLOT`), phong cách biểu đồ (`PLOT_STYLE`), và giới hạn số bins tối đa (`MAX_FD_BINS`).

Định nghĩa hàm hỗ trợ

- `ensure_dir`: Đảm bảo thư mục đầu ra tồn tại.
- `sanitize_filename`: Làm sạch tên chỉ số thống kê để tạo tên file hợp lệ.
- `calculate_fd_bins`: Tính số lượng bins tối ưu cho histogram dựa trên quy tắc Freedman-Diaconis (ưu tiên) hoặc Sturges (dự phòng), giúp biểu đồ phản ánh tốt hơn cấu trúc dữ liệu thực tế và tránh việc chọn số bins tùy tiện. Có giới hạn số bins tối đa để tránh quá chi tiết.

Định nghĩa hàm vẽ biểu đồ

- `plot_overall_hist`:
  - Nhận DataFrame, tên chỉ số thống kê, và thư mục đầu ra.
  - Lọc dữ liệu cho chỉ số thống kê, chuyển đổi sang dạng số và loại bỏ giá trị thiếu (`NaN`).
  - Tính số bins tối ưu bằng `calculate_fd_bins`.
  - Sử dụng `seaborn.histplot` để vẽ histogram phân phối tổng thể, bật KDE.
  - Đặt tiêu đề, nhãn trục và lưu biểu đồ vào file PNG với tên được chuẩn hóa.
- `plot_team_hist`:
  - Nhận DataFrame, tên chỉ số, thư mục đầu ra, tên cột đội, và số đội trên mỗi hình.
  - Lấy danh sách các đội duy nhất và chia thành các nhóm nhỏ (theo `TEAMS_PER_PLOT`).
  - Lặp qua từng nhóm đội:
    - Tạo một figure mới với các subplots (lưới biểu đồ con).
    - Lọc dữ liệu chỉ chứa các đội trong nhóm hiện tại.
    - Tính số bins tối ưu *dựa trên phạm vi dữ liệu của các đội trong nhóm này*. Điều này giúp các histogram trong cùng một figure có cùng cách chia khoảng giá trị.
    - Xác định giới hạn trục x (`x_min`, `x_max`) chung cho tất cả các subplot trong figure hiện tại để dễ so sánh.
    - Vẽ histogram cho từng đội trên một subplot riêng biệt bằng `seaborn.histplot`, sử dụng số bins và phạm vi bin (`binrange`) đã tính.
    - Đặt tiêu đề con là tên đội, ẩn nhãn trục để tránh rối mắt, giới hạn trục x thống nhất.
    - Ẩn các subplot thừa.
    - Đặt tiêu đề chính cho figure (bao gồm tên chỉ số và nhóm đội).
    - Lưu figure chứa histogram của nhóm đội vào file PNG.

Hàm thực thi chính (`if __name__ == '__main__':`)

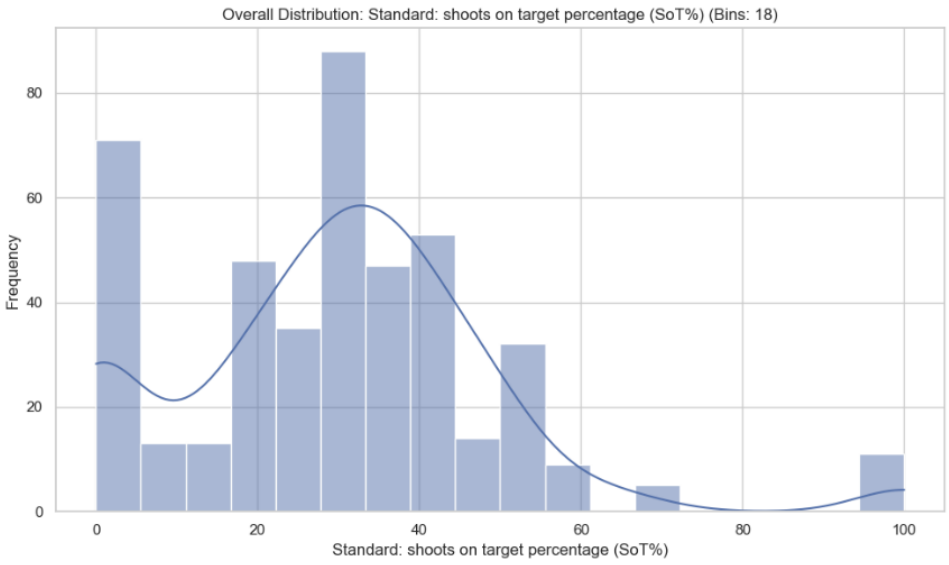
- Thiết lập theme cho `seaborn`.
- Định nghĩa danh sách các chỉ số cần vẽ (`stats_to_plot`).
- Xác định đường dẫn và đọc file `results.csv` vào DataFrame.
- Đảm bảo thư mục đầu ra tồn tại.
- Lặp qua từng chỉ số trong `stats_to_plot`:
  - Gọi `plot_overall_hist` để vẽ biểu đồ tổng thể.
  - Gọi `plot_team_hist` để vẽ biểu đồ cho từng đội (được nhóm lại).

2.3.4 Kết quả trực quan

Kết quả đầu ra của script là một tập hợp các file ảnh PNG được lưu trong thư mục `stat_histograms`. Cụ thể:

- Với mỗi chỉ số thống kê trong `stats_to_plot`, sẽ có *một file* biểu đồ histogram thể hiện phân phối tổng thể của chỉ số đó trên toàn giải đấu

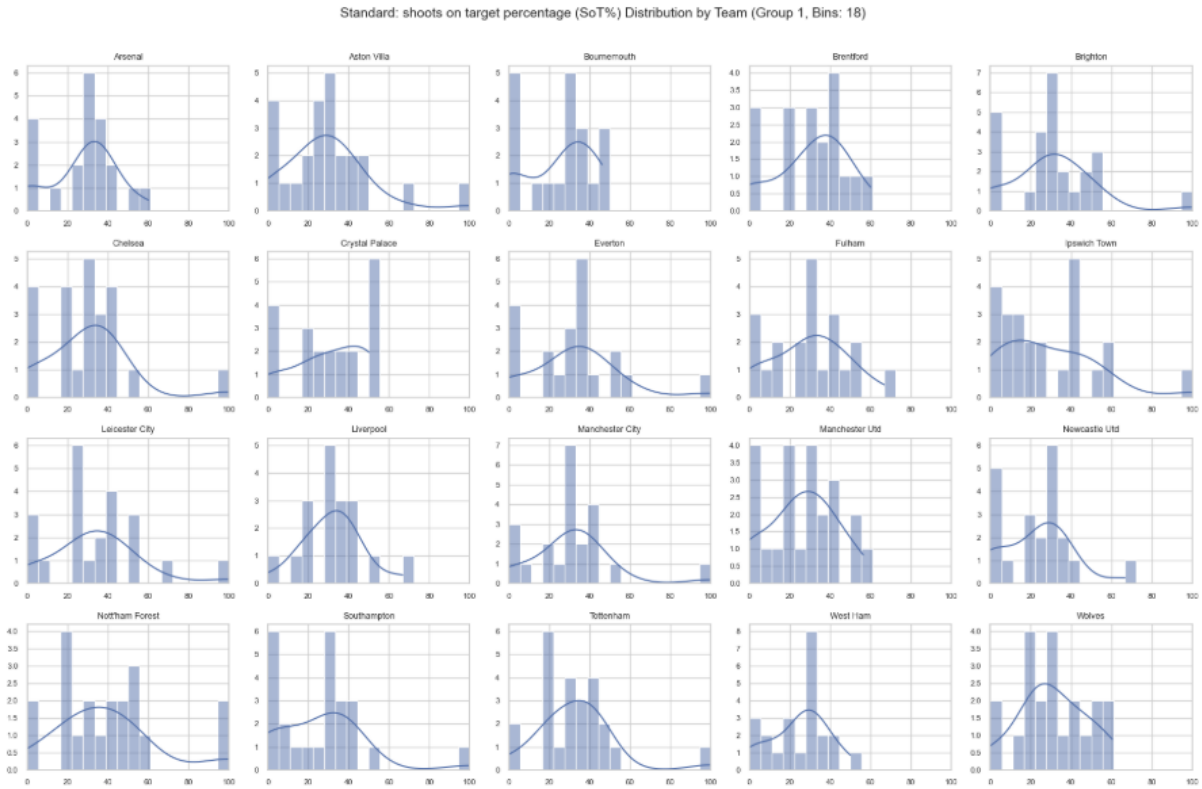
  (ví dụ: `hist_overall_Standard_shoots_on_target_percentage_SoT.png`).



Hình 2.3: Ví dụ biểu đồ histogram tổng thể (Nguồn: Trích từ tài liệu gốc).

- Với mỗi chỉ số thống kê, sẽ có *một hoặc nhiều file* biểu đồ histogram thể hiện phân phối của chỉ số đó theo từng đội. Các đội được nhóm lại (mặc định 20 đội/hình) để tránh tạo quá nhiều file hoặc hình ảnh quá lớn

  (ví dụ: `hist_teams_tandard_shoots_on_target_percentage_SoT_group_1.png`,...).



Hình 2.4: Ví dụ biểu đồ histogram theo nhóm đội (Nguồn: Trích từ tài liệu gốc).

Mỗi file này chứa một lưới các biểu đồ nhỏ, mỗi biểu đồ nhỏ tương ứng với một đội. Các biểu đồ này cho phép người xem quan sát trực quan hình dạng phân phối (đối xứng, lệch, đa đỉnh), phạm vi giá trị, và tần suất xuất hiện của các giá trị khác nhau cho từng chỉ số, cả trên tổng thể và trong từng đội.

2.3.5 Hạn chế

Trong quá trình phân tích dữ liệu thống kê cầu thủ, biểu đồ histogram được sử dụng để trực quan hóa phân phối của từng chỉ số (ví dụ: số bàn thắng, số phút thi đấu, số pha kiến tạo, v.v.). Đây là công cụ hiệu quả giúp quan sát nhanh xu hướng phổ biến, phát hiện các giá trị ngoại lai (outliers) và so sánh sự khác biệt giữa các cầu thủ hoặc đội bóng. Tuy nhiên, việc sử dụng histogram cũng tồn tại một số điểm hạn chế cần được lưu ý để tránh đưa ra các kết luận thiếu chính xác hoặc phiến diện:

- **Phụ thuộc vào chất lượng dữ liệu:** Kết quả trực quan hoàn toàn phụ thuộc vào tính đầy đủ và chính xác của dữ liệu trong file `results.csv`. Các giá trị thiếu (`NaN`) đã bị loại bỏ trong quá trình xử lý để đảm bảo tính chính xác của biểu đồ. Tuy nhiên, nếu tỷ lệ dữ liệu thiếu lớn hoặc không phân bố ngẫu nhiên, điều này có thể làm sai lệch hình dạng phân phối và dẫn đến nhận định thiếu khách quan về xu hướng tổng thể.

- **Nhạy cảm với số lượng Bins:** Mặc dù số lượng bins được xác định dựa trên quy tắc Freedman-Diaconis nhằm tối ưu hóa mức độ chi tiết của biểu đồ, histogram vẫn có thể thay đổi hình dạng nếu thay đổi số lượng bins. Bên cạnh đó, việc đặt giới hạn tối đa cho số bins (qua biến `MAX_FD_BINS`) mang tính chủ quan và có thể ảnh hưởng đến mức độ phân giải của biểu đồ, đặc biệt trong các phân phối có nhiều giá trị phân tán.

2.4 Phân tích hiệu suất các đội bóng Premier League 2024–2025

2.4.1 Giới thiệu

Nhằm đánh giá hiệu suất thi đấu của các đội bóng tại mùa giải 2024–2025, quá trình phân tích được thiết kế với hai mục tiêu chính: xác định đội dẫn đầu ở từng chỉ số thống kê và tìm ra đội có hiệu suất tổng thể tốt nhất. Việc xử lý dữ liệu được thực hiện thông qua hai tệp chính:

- `highest_stats_team.py`: Tệp mã nguồn Python đảm nhận toàn bộ quy trình phân tích, từ việc đọc dữ liệu, phân loại các chỉ số (tốt, xấu, bỏ qua), xác định đội dẫn đầu theo từng tiêu chí, đến tổng hợp kết quả để xác định đội xuất sắc nhất mùa giải.
- `config.json`: Tệp cấu hình cho phép tùy chỉnh linh hoạt các tham số như đường dẫn dữ liệu, cách xử lý giá trị thiếu và hệ thống phân loại chỉ số. Việc tách riêng cấu hình giúp điều chỉnh tiêu chí phân tích mà không cần sửa đổi mã nguồn.

Cách tiếp cận này giúp tối ưu hóa khả năng tái sử dụng, dễ bảo trì và mở rộng cho các mùa giải hoặc bộ dữ liệu khác nhau.

2.4.2 Ý tưởng chính

Xác định đội có điểm cao nhất cho mỗi chỉ số: script phân loại các chỉ số thành ba nhóm (tốt, xấu, bỏ qua) dựa trên file cấu hình (`config.json`). Với mỗi chỉ số, code chuyển đổi dữ liệu thành dạng số, sau đó tìm giá trị tối ưu (cao nhất cho chỉ số tốt, thấp nhất cho chỉ số xấu) và xác định đội nào đạt được giá trị đó. Ví dụ, đội có nhiều bàn thắng nhất, đội có tỷ lệ chuyền bóng thành công cao nhất, hoặc đội có ít thẻ đỏ nhất.

Xác định đội có hiệu suất tổng thể tốt nhất: Code đếm số lần mỗi đội dẫn đầu ở các chỉ số khác nhau, tính riêng cho chỉ số tốt và chỉ số xấu. Sau đó, tính tổng điểm kết hợp cho mỗi đội bằng cách cộng số lần dẫn đầu ở cả hai loại chỉ số. Đội có tổng số lần dẫn đầu nhiều nhất được coi là đội có hiệu suất tổng thể tốt nhất trong mùa giải.

2.4.3 Quy trình thực thi

Quy trình phân tích được điều khiển bởi tập lệnh `highest_stats_team.py`, với các bước chính sau:

Khởi tạo & tải cấu hình

Tải cấu hình từ `config.json` bằng hàm `load_config`. Hàm này đọc tệp JSON và tạo các tập hợp (set) từ khóa `ignore_stats_set`, `bad_stats_set`, `good_stats_set` để phân loại chỉ số.

Xác định đường dẫn & đọc dữ liệu CSV

Xây dựng đường dẫn tuyệt đối đến tệp `results2.csv` (nằm trong `../calculating_statistics/`). Đọc dữ liệu từ CSV vào DataFrame `dataframe` bằng `pandas`, sử dụng `na_values` từ config.

Tiền xử lý DataFrame

Xác định cột tên đội (mặc định là cột đầu tiên). Chuẩn hóa cột tên đội: chuyển sang chữ thường, loại bỏ các hàng có tên đội là "all".

Xác định đội dẫn đầu cho mỗi chỉ số thống kê hợp lệ (Hàm `analyze_performance_by_stat_type`)

Bước 1: Xác định và Lọc Chỉ số cần Phân tích:

- Tạo danh sách các cột có tiền tố "Mean of "(trừ cột đội).
- Với mỗi cột tiềm năng:
  - Trích xuất tên chỉ số thực (bỏ "Mean of ").
  - Phân loại chỉ số (dùng `classify_stat`) thành 'ignore', 'bad', 'good', hoặc 'uncategorized' dựa trên các tập hợp từ khóa đã tải.
  - Nếu không phải 'ignore', chuyển cột sang dạng số. Chỉ giữ lại các chỉ số có ít nhất một giá trị số hợp lệ.

Bước 2: Phân tích từng Chỉ số đã Lọc:

- Với mỗi chỉ số được chọn để phân tích:
  - Chuyển cột chỉ số sang dạng số và loại bỏ các giá trị không hợp lệ (`NaN`).
  - Tìm giá trị tốt nhất:
    - Nếu loại là 'good' hoặc 'uncategorized': tìm giá trị *lớn nhất*.
    - Nếu loại là 'bad': tìm giá trị *nhỏ nhất*.
  - Nếu không tìm thấy giá trị tốt nhất (ví dụ: cột rỗng), ghi nhận lỗi.
  - Ngược lại:
    - Xác định (các) đội đạt được giá trị tốt nhất này.
    - Lưu tên (các) đội dẫn đầu, giá trị tốt nhất (ở dạng gốc để hiển thị) và loại chỉ số vào một dictionary kết quả.

Bước 3: Trả về: Dictionary chứa kết quả phân tích cho từng chỉ số.

Hiển thị kết quả phân tích chi tiết

Sắp xếp kết quả theo tên chỉ số. Lặp qua từng chỉ số đã phân tích, định dạng giá trị (dùng `format_value` để xử lý số nguyên, số thực, "N/A") và in ra: tên chỉ số, loại (GOOD, BAD), đội dẫn đầu, giá trị đạt được.

Tính toán và hiển thị bảng xếp hạng tổng Hợp

Bước 1: Đếm số lần dẫn đầu:

- Với mỗi chỉ số có kết quả hợp lệ trong `analysis_results`:
  - Nếu đội dẫn đầu chỉ số 'good', tăng điểm cho đội đó trong `good_lead_counts`.
  - Nếu đội dẫn đầu chỉ số 'bad', tăng điểm cho đội đó trong `bad_lead_counts`.

Bước 2: Tạo điểm tổng hợp:

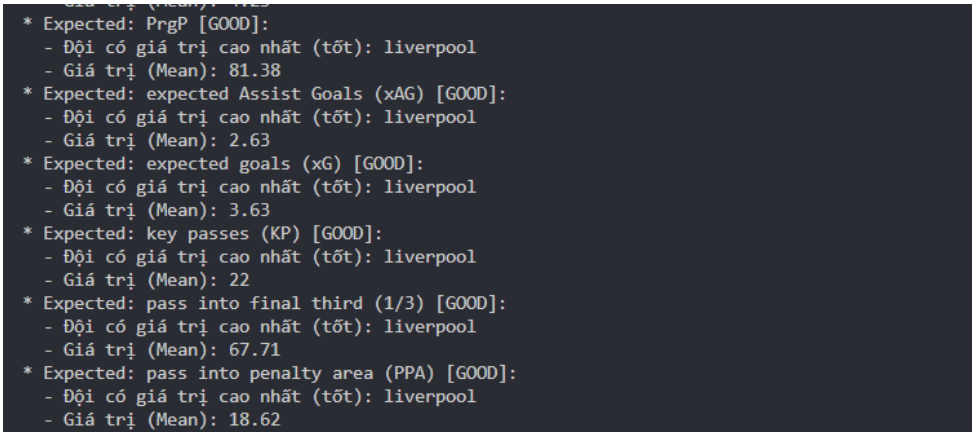
- Lấy danh sách tất cả các đội duy nhất từ DataFrame.
- Khởi tạo điểm 0 cho tất cả các đội.
- Cộng tổng số lần dẫn đầu 'good' và 'bad' để có điểm tổng hợp cho mỗi đội.

Bước 3: Sắp xếp và Hiển thị:

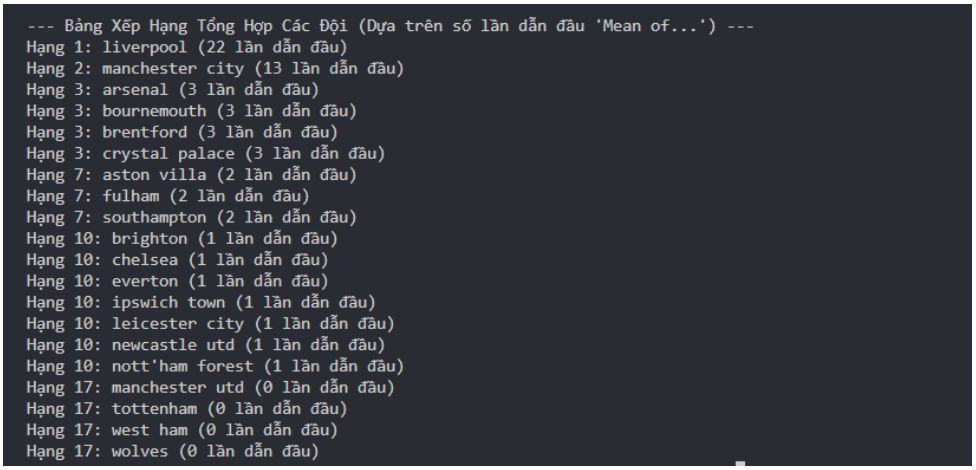
- Sắp xếp các đội: giảm dần theo điểm tổng hợp, sau đó tăng dần theo tên đội (nếu cùng điểm).
- In bảng xếp hạng với thứ hạng, tên đội và tổng số lần dẫn đầu.

2.4.4 Kết quả

Khi script `highest_stats_team.py` được thực thi thành công, nó sẽ tạo ra hai khối thông tin chính được in ra console (màn hình dòng lệnh). Các kết quả này cung cấp một cái nhìn chi tiết về hiệu suất của từng đội đối với mỗi chỉ số thống kê được phân tích, và một bảng xếp hạng tổng hợp các đội dựa trên số lần họ dẫn đầu các chỉ số đó.



Hình 2.5: Ví dụ kết quả phân tích chi tiết theo chỉ số.



Hình 2.6: Ví dụ bảng xếp hạng tổng hợp các đội.

Dữ liệu thống kê cho thấy Liverpool là một tập thể thi đấu vượt trội và toàn diện, không chỉ nổi bật ở một vài khía cạnh mà duy trì phong độ ấn tượng trên hầu hết các chỉ số quan trọng của bóng đá hiện đại. Họ sở hữu sức mạnh tấn công bùng nổ, khả năng

kiểm soát bóng vượt trội cùng hệ thống phòng ngự vững chắc – tất cả tạo nên một cỗ máy thi đấu trơn tru và hiệu quả. Trên thực tế họ đã giành được chức vô địch Premier League mùa giải 2024–2025 *trước 4 vòng đấu* một cách đầy thuyết phục, với *82 điểm* – bỏ xa đội nhì bảng Arsenal một khoảng cách an toàn. Đây chính là minh chứng rõ ràng rằng, khi một đội bóng vận hành đúng hướng cả về chuyên môn lẫn chiến lược, sự áp đảo của họ sẽ được thể hiện rõ rệt qua các con số thống kê và kết quả trên sân cỏ.

Chương 3

Phân cụm cầu thủ và giảm chiều dữ liệu với PCA

3.1 Phân cụm cầu thủ bóng đá sử dụng thuật toán k-means

3.1.1 Giới thiệu

Phân tích dữ liệu cầu thủ đóng vai trò quan trọng trong thể thao hiện đại, giúp các huấn luyện viên, nhà phân tích và người quản lý đưa ra quyết định sáng suốt. Một trong những kỹ thuật phổ biến được sử dụng là thuật toán phân cụm K-means, nhằm mục đích phân chia một tập hợp các cầu thủ thành các nhóm (cụm) riêng biệt dựa trên sự tương đồng về các chỉ số thống kê của họ. Phần này sẽ trình bày cách thuật toán K-means được áp dụng, đặc biệt tập trung vào việc xác định số lượng cụm tối ưu và bình luận về kết quả thu được từ việc chạy tệp mã nguồn `player_clustering_kmeans.py`.

3.1.2 Lựa chọn phạm vi k để chạy thử nghiệm

Việc chọn phạm vi k để chạy thử nghiệm trong bài toán phân cụm cầu thủ (500 cầu thủ, 78 chỉ số) cần dựa trên các yếu tố lý thuyết, đặc điểm dữ liệu, và ngữ cảnh thực tiễn trong bóng đá.

Lý do chọn kmin = 3

- **Yêu cầu tối thiểu để có phân cụm có ý nghĩa:**
  - k=1: không tạo ra cụm (tất cả dữ liệu thuộc cùng một nhóm), không có giá trị phân tích.
  - k=2: chỉ chia dữ liệu thành hai nhóm, thường quá đơn giản cho dữ liệu khoảng 78 chỉ số (chỉ phân biệt thô sơ như "tấn công"và "phòng ngự").
  - k=3: điểm khởi đầu hợp lý để phản ánh các vai trò cơ bản trong bóng đá, ví dụ: thủ môn, cầu thủ phòng ngự, cầu thủ tấn công.
- **Phù hợp với số lượng mẫu:**
  - Với 500 cầu thủ, k=3 tạo ra các cụm trung bình $500/3 \approx 167$ cầu thủ/cụm, đủ lớn để mang ý nghĩa thống kê.

Kết luận: kmin=3 là điểm khởi đầu hợp lý.

Lý do chọn kmax = 15

- **Dựa trên số lượng mẫu (khoảng 500 cầu thủ):**
  - Mỗi cụm nên có ít nhất 20–50 mẫu để đảm bảo ý nghĩa thống kê:
    - Nếu mỗi cụm có tối thiểu 20 cầu thủ: $k = 500/20 = 25$
    - Nếu mỗi cụm có tối thiểu 50 cầu thủ: $k = 500/50 = 10$.
  - Để các cụm không quá nhỏ, kmax nên nằm trong khoảng 10–25. Tuy nhiên, cần cân nhắc thêm ngữ cảnh.
- **Ngữ cảnh bóng đá:**
  - Số vai trò/phong cách chơi trong bóng đá hiếm khi vượt quá 15–20 (ví dụ: thủ môn, hậu vệ trung tâm, hậu vệ cánh, tiền vệ phòng ngự, tiền vệ tấn công, tiền vệ cánh, trung phong, tiền đạo lùi, v.v.).
  - k=15 đủ để bao quát các vai trò chi tiết mà không vượt quá giới hạn thực tế.
- **Kinh nghiệm thực tiễn:**
  - Trong các bài toán phân cụm tương tự, kmax=15 thường đủ để bao quát các điểm khuỷu tay (thường xuất hiện trong khoảng k=4 đến k=10), tránh lãng phí tài nguyên tính toán.

Kết luận: kmax=15 là ngưỡng hợp lý. Phạm vi k từ 3 đến 15 là hợp lý: kmin=3 tạo cụm cơ bản (167 cầu thủ/cụm), kmax=15 bao quát vai trò chi tiết (33 cầu thủ/cụm), phù hợp với dữ liệu, ngữ cảnh bóng đá, và điểm khuỷu tay thường trong k=4 đến k=10.

3.1.3 Xác định số lượng cụm tối ưu (K) và Quy trình thực thi

Ý tưởng chính: Để xác định số lượng cụm (K) tối ưu cho việc phân nhóm cầu thủ, em đã áp dụng phương pháp Elbow. Quy trình này bắt đầu bằng việc thử nghiệm thuật toán K-Means với một loạt các giá trị K tiềm năng, cụ thể là từ 3 đến 15 cụm. Đối với mỗi giá trị K, chỉ số WCSS (Within-Cluster Sum of Squares - tổng bình phương khoảng cách từ mỗi điểm dữ liệu đến tâm cụm của nó) được tính toán. WCSS là một thước đo về độ chặt chẽ của các cụm; giá trị WCSS thấp hơn cho thấy các điểm dữ liệu trong cùng một cụm gần nhau hơn. Sau đó, các giá trị WCSS này được trực quan hóa trên một biểu đồ đường, với số lượng cụm (K) trên trục hoành và WCSS tương ứng trên trục tung. Điểm "khuỷu tay"(elbow point) trên biểu đồ, nơi mà việc tăng thêm số cụm K không còn dẫn đến sự sụt giảm đáng kể của WCSS, được xác định là giá trị K tối ưu. Dựa trên phân tích biểu đồ Elbow, một giá trị K phù hợp đã được lựa chọn. Cuối cùng, thuật toán K-Means được thực thi lại với số cụm K tối ưu này để tiến hành phân chia các cầu thủ vào các nhóm tương ứng, qua đó mỗi cầu thủ được gán vào một cụm dựa trên sự tương đồng về các chỉ số thống kê.

Cấu hình và khởi tạo

- **Import thư viện:** Nạp các thư viện cần thiết như `pandas`, `numpy`, `os`, `sklearn.cluster.KMeans` (cho thuật toán K-Means), `sklearn.preprocessing.StandardScaler` (để chuẩn hóa dữ liệu), `sklearn.impute.SimpleImputer` (để xử lý giá trị thiếu), `matplotlib.pyplot` và `seaborn` (để vẽ biểu đồ).

- **Định nghĩa hằng số cấu hình:**
  - `PLAYER_COL`: Tên cột chứa tên cầu thủ (mặc định là 'Name').
  - `CSV_RELATIVE_PATH`: Đường dẫn tương đối đến tệp CSV chứa dữ liệu cầu thủ (mặc định là `../problem_1/results1.csv`).
  - `script_dir`: Lấy đường dẫn thư mục hiện tại của script.
  - `output_folder_name`: Tên thư mục để lưu kết quả phân tích (mặc định là `kmeans_analysis_results`).
  - `OUTPUT_DIR`: Tạo đường dẫn tuyệt đối đến thư mục lưu kết quả.
- **Định nghĩa hàm trợ giúp** `ensure_dir(path: str)`**:** Hàm này đảm bảo rằng một thư mục tồn tại tại đường dẫn được cung cấp; nếu không, nó sẽ tạo thư mục đó.

Hàm chính `cluster_players_kmeans(csv_path: str, player_col: str, max_k: int = 10)`

- **Nạp Dữ liệu:**
  - Xác định đường dẫn tuyệt đối đến tệp CSV.
  - Đọc dữ liệu từ tệp CSV vào một DataFrame của `pandas`.
- **Chuẩn bị Dữ liệu:**
  - Chọn các cột số: Xác định và chọn tất cả các cột trong DataFrame có kiểu dữ liệu là số để sử dụng làm đặc trưng cho việc phân cụm (loại bỏ cả cột 'Age' để làm cho các cụm phản ánh rõ ràng hơn sự tương đồng về kỹ năng, phong cách chơi hoặc vai trò chiến thuật, thay vì bị ảnh hưởng bởi yếu tố giai đoạn sự nghiệp của cầu thủ).
  - Xử lý giá trị thiếu (NaN): Sử dụng `SimpleImputer` để thay thế bất kỳ giá trị thiếu nào trong các cột số đã chọn bằng giá trị trung bình của cột tương ứng.
  - Chuẩn hóa Dữ liệu: Sử dụng `StandardScaler` để chuẩn hóa các đặc trưng số. Việc này đảm bảo rằng tất cả các đặc trưng đều có cùng một thang đo, điều này rất quan trọng đối với các thuật toán dựa trên khoảng cách như K-Means.
- **Xác định số cụm tối ưu (K) bằng phương pháp Elbow:**
  - Khởi tạo một danh sách `wcss` (Within-Cluster Sum of Squares - tổng bình phương khoảng cách nội cụm).
  - Lặp qua một phạm vi các giá trị K có thể (từ 3 đến 15).
  - Với mỗi giá trị K:
    - Tạo một mô hình K-Means (`KMeans`) với số cụm là K, phương thức khởi tạo `'k-means++'`, `n_init='auto'` (để tự động chọn số lần chạy với các hạt giống tâm cụm khác nhau), và `random_state=42` (để đảm bảo kết quả có thể tái tạo).
    - Huấn luyện mô hình K-Means trên dữ liệu đã chuẩn hóa.
    - Lấy giá trị `inertia_` (WCSS) của mô hình và thêm vào danh sách `wcss`.
  - Vẽ biểu đồ Elbow:
    - Gọi hàm `ensure_dir` để tạo thư mục đầu ra nếu chưa có.

    - Tạo một biểu đồ đường (`plot`) với số lượng cụm (K) trên trục hoành và WCSS trên trục tung.
    - Đặt tiêu đề, nhãn trục, và hiển thị lưới cho biểu đồ.
    - Lưu biểu đồ Elbow dưới dạng tệp PNG vào thư mục `OUTPUT_DIR`.
  - In ra thông báo hướng dẫn người dùng xem biểu đồ và chọn giá trị K tối ưu (điểm "khuỷu tay"trên biểu đồ, nơi WCSS bắt đầu giảm chậm lại).

- **Nhận đầu vào K tối ưu từ người dùng:**
  - Yêu cầu người dùng nhập số cụm K tối ưu dựa trên biểu đồ Elbow.
  - Vòng lặp sẽ tiếp tục cho đến khi người dùng nhập một số nguyên hợp lệ trong phạm vi cho phép.

- **Áp dụng K-means với K tối ưu:**
  - Sau khi người dùng chọn K, tạo một mô hình K-Means cuối cùng với số cụm `optimal_k` đã chọn.
  - Huấn luyện mô hình K-Means cuối cùng này trên dữ liệu đã chuẩn hóa.
  - Lấy nhãn cụm (cluster labels) được gán cho mỗi điểm dữ liệu (mỗi cầu thủ).

- **Thêm nhãn cụm vào DataFrame gốc:**
  - Tạo một bản sao của DataFrame gốc.
  - Thêm một cột mới có tên `'Cluster'` vào bản sao này, chứa các nhãn cụm đã được gán.

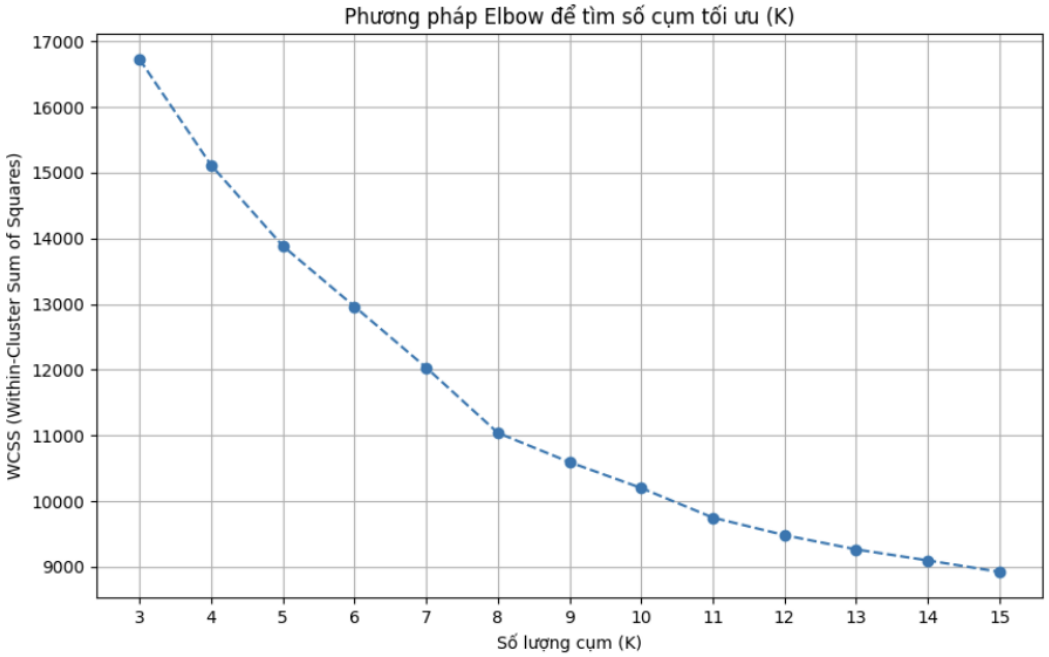
- **Phân tích và Lưu kết quả:**
  - Phân tích tâm cụm:
    - Tính toán tọa độ tâm của mỗi cụm trên thang đo đã chuẩn hóa (`final_kmeans.cluster_centers_`).
    - Chuyển đổi các tọa độ tâm này trở lại thang đo gốc bằng cách sử dụng `scaler.inverse_transform()`.
    - Tạo một DataFrame mới (`centroid_df`) chứa các giá trị trung bình của các đặc trưng số cho mỗi cụm (ở thang đo gốc).
    - In ra DataFrame tâm cụm này (đã làm tròn đến 2 chữ số thập phân).
    - Lưu DataFrame tâm cụm vào một tệp CSV trong thư mục `OUTPUT_DIR`.
  - Đếm số lượng cầu thủ trong mỗi cụm: In ra số lượng cầu thủ thuộc về mỗi cụm.
  - Sắp xếp và Lưu DataFrame đã phân cụm:
    - Sắp xếp DataFrame đã được gán nhãn cụm theo cột `'Cluster'`.
    - Lưu DataFrame đã sắp xếp này vào một tệp CSV khác trong thư mục `OUTPUT_DIR`.

- **Trả về** DataFrame đã phân cụm, đối tượng `scaler`, DataFrame các đặc trưng đã chuẩn hóa, và danh sách các cột số.

Khối thực thi chính (`if __name__ == '__main__':`)

- **Gọi hàm phân cụm:** Gọi hàm `cluster_players_kmeans` với các tham số đã cấu hình (`CSV_RELATIVE_PATH`, `PLAYER_COL`).
- **Lưu trữ các kết quả trả về** (DataFrame đã phân cụm, scaler, v.v.).
- **Hiển thị ví dụ cầu thủ từ mỗi cụm (nếu phân cụm thành công):**
  - Kiểm tra xem `clustered_data` có tồn tại không.
  - Nếu có và danh sách `numeric_cols_list` không rỗng:
    - Lặp qua từng ID cụm duy nhất (đã sắp xếp).
    - In ra tiêu đề cho cụm đó.
    - Lấy 5 cầu thủ đầu tiên (`head(5)`) từ cụm hiện tại.
    - Chọn các cột để hiển thị: tên cầu thủ (`PLAYER_COL`), tối đa 5 đặc trưng số đầu tiên, và cột `'Cluster'`.
    - In ra thông tin của các cầu thủ mẫu này.

3.1.4 Kết quả phân cụm K-means



Hình 3.1: Phương pháp Elbow để tìm số cụm tối ưu (K).

Biểu đồ Elbow cho thấy WCSS giảm mạnh từ k = 3 (17000) đến k = 6 (12000), sau đó giảm chậm từ k = 6 đến k = 15 (xuống 9000), đánh dấu k = 6 là điểm "cùi chỏ". Lựa chọn k = 6 hợp lý vì nó cân bằng giữa chất lượng phân cụm (WCSS đã giảm đáng kể) và độ phức tạp mô hình, tránh tạo thêm cụm không cần thiết khi WCSS sau đó giảm không đáng kể. Số lượng cầu thủ trong mỗi cụm sẽ được in ra màn hình.

```
Số lượng cầu thủ trong mỗi cụm:
Cluster
0    148
1     31
2     97
3    126
4     50
5     39
Name: count, dtype: int64
```

Hình 3.2: Số lượng cầu thủ trong mỗi cụm.

Các giá trị trung bình của từng chỉ số trong mỗi cụm được lưu vào tệp `cluster_centroids.csv`, trong khi kết quả phân cụm chi tiết của từng cầu thủ được lưu vào tệp `player_clusters_sorted.csv`. Dựa trên dữ liệu từ hai tệp này, em đã tiến hành phân tích đặc điểm nổi bật của từng cụm như sau:

- **Cụm 0 (148 cầu thủ, 30.1%): Cầu thủ dự bị có vai trò phòng ngự hoặc Thủ môn.**
  - Đặc trưng: Thời gian thi đấu thấp, đóng góp tấn công gần như không có (bàn thắng, kiến tạo, sút bóng, rê dắt tấn công đều ở mức thấp nhất trong các cụm).
  - Hoạt động chủ yếu ở phần sân nhà (`Touches: Def Pen` cao), tỷ lệ chuyền ngắn/trung bình tốt (phù hợp với việc phát bóng hoặc luân chuyển an toàn).
  - Số lượng: Cụm lớn nhất, cho thấy đây là nhóm phổ biến bao gồm các cầu thủ ít ra sân, chuyên gia phòng ngự sâu, hoặc có khả năng cao là các thủ môn của nhiều đội.
- **Cụm 1 (31 cầu thủ, 6.3%): Tiền đạo mục tiêu/Sát thủ vòng cấm.**
  - Đặc trưng: Ghi bàn (`Performance: goals`) và chỉ số bàn thắng kỳ vọng (`Expected: expected goals (xG)`) cao nhất tuyệt đối.
  - Là điểm đến của các đường chuyền tấn công (`Progression: PrgR` cao nhất) và hoạt động nhiều nhất trong vòng cấm đối phương (`Touches: Att Pen` cao nhất).
  - Khả năng tự phát triển bóng bằng chuyền (`Progression: PrgP`) thấp, củng cố vai trò người dứt điểm cuối cùng.
  - Số lượng: Cụm nhỏ nhất, phản ánh sự chuyên biệt của các chân sút có hiệu suất cao và vai trò rõ ràng như vậy.
- **Cụm 2 (97 cầu thủ, 19.8%): Tiền vệ trung tâm đa năng, cân bằng công thủ.**
  - Đặc trưng: Thời gian thi đấu cao. Dẫn đầu về các chỉ số phòng ngự quan trọng (ví dụ: `Tackles: Tkl` cao nhất, `Performance: Recov` cao nhất).
  - Phát triển bóng hiệu quả từ tuyến hai (`Progression: PrgP` cao, `Expected: pass into final third (1/3)` nhiều) nhưng không phải là người kiến tạo chính yếu như Cụm 4 (kiến tạo, key passes thấp hơn đáng kể).
  - Số lượng: Cụm lớn, cho thấy tầm quan trọng và sự phổ biến của các tiền vệ "xương sống", quán xuyến khu trung tuyến và kết nối lối chơi.

- **Cụm 3 (126 cầu thủ, 25.7%): Cầu thủ dự bị có thiên hướng tấn công, tạo đột biến.**
  - Đặc trưng: Thời gian thi đấu thấp nhất nhưng hiệu suất tấn công tính trên 90 phút rất đáng chú ý (đặc biệt `SCA: SCA90` - hành động tạo cú sút/90 phút - rất cao).
  - Đóng góp phòng ngự thấp, cho thấy xu hướng được sử dụng để làm mới hàng công.
  - Số lượng: Cụm lớn thứ hai, phản ánh việc các đội thường có nhiều phương án dự phòng để thay đổi cục diện trận đấu bằng những cầu thủ có khả năng tấn công.
- **Cụm 4 (50 cầu thủ, 10.2%): Tiền vệ tấn công/Tiền đạo cánh kiến thiết chủ lực.**
  - Đặc trưng: Dẫn đầu tuyệt đối ở hầu hết các chỉ số sáng tạo và kiến thiết lối chơi (`Performance: assists`, `Expected: key passes (KP)`, `SCA: SCA`, `GCA: GCA` đều cao nhất).
  - Khả năng rê dắt (`Progression: PrgC`, `Take-Ons: Att`) và chuyền bóng phát triển tấn công (`Progression: PrgP`) vượt trội.
  - Đồng thời cũng có khả năng ghi bàn tốt.
  - Số lượng: Số lượng cầu thủ ở mức vừa phải, phản ánh vai trò quan trọng của các "nhạc trưởng"tấn công, những người thường là nhân tố chủ chốt nhưng không phải lúc nào cũng chiếm số đông trong một đội hình.
- **Cụm 5 (39 cầu thủ, 7.9%): Trung vệ chủ chốt, có khả năng phát triển bóng từ tuyến dưới.**
  - Đặc trưng: Thời gian thi đấu cao nhất. Thống trị các chỉ số phòng ngự chuyên biệt (`Blocks: Blocks`, `Blocks: Int`, `Aerial Duels: Won` đều cao nhất).
  - Tỷ lệ chuyền bóng thành công cao nhất (`Total: Pass completion (Cmp%)`) và khối lượng chuyền tịnh tiến từ sân nhà rất lớn (`Progression: PrgP`, `Expected: pass into final third (1/3)` cao), nhưng đóng góp tấn công trực tiếp (bàn thắng, kiến tạo) cực kỳ thấp.
  - Số lượng: Cụm nhỏ, phù hợp với số lượng trung vệ đá chính thường có trong một đội bóng, đặc biệt là những người có bộ kỹ năng toàn diện cả phòng ngự lẫn phát triển bóng.

3.2 Trực quan hóa kết quả phân cụm cầu thủ bằng phương pháp phân tích thành phần chính (PCA)

3.2.1 Giới thiệu

Bài tập này nhằm mục tiêu trực quan hóa kết quả phân cụm các cầu thủ bằng cách sử dụng phương pháp Phân Tích Thành Phần Chính (PCA). Dữ liệu đầu vào được lấy từ kết quả phân cụm trước đó, vốn được thực hiện bởi module `player_clustering_kmeans.py` dựa trên tập dữ liệu `results.csv`. Đầu tiên, các đặc trưng của cầu thủ đã được chuẩn hóa và sau đó được giảm số chiều từ nhiều chiều xuống còn hai thành phần chính thông

qua kỹ thuật PCA. Quá trình này cho phép biểu diễn dữ liệu một cách trực quan trên mặt phẳng hai chiều, đồng thời vẫn giữ lại phần lớn thông tin quan trọng về sự khác biệt giữa các cụm. Các cầu thủ sau đó được hiển thị trên biểu đồ phân tán 2D, với mỗi điểm dữ liệu được tô màu theo nhãn cụm đã được gán trước đó từ thuật toán K-Means. Kết quả cuối cùng bao gồm biểu đồ trực quan hóa được lưu lại thành hình ảnh, cùng với bộ dữ liệu đã giảm chiều và nhãn phân cụm đi kèm, giúp thuận tiện cho việc phân tích, đối chiếu và trình bày sau này.

3.2.2 Quy trình thực hiện (Dựa trên script `pca_visual_kmeans.py`)

Đoạn mã `pca_visual_kmeans.py` thực hiện các bước chính sau:

Bước 1: Nạp và xử lý kết quả phân cụm K-Means:

- Script gọi hàm `cluster_players_kmeans` từ module `player_clustering_kmeans.py`.
- Hàm này chịu trách nhiệm đọc dữ liệu cầu thủ từ `CSV_RELATIVE_PATH` (trỏ đến `../problem_1/results.csv`), thực hiện phân cụm K-Means (bao gồm việc xác định số cụm `optimal_k` và chuẩn hóa dữ liệu).
- Kết quả trả về bao gồm DataFrame `df_clustered_sorted` (chứa thông tin cầu thủ và nhãn cụm `Cluster` đã gán) và `features_scaled_df` (DataFrame các đặc trưng số đã được chuẩn hóa, là đầu vào cho PCA).

Bước 2: Áp dụng phân tích thành phần chính (PCA):

- Một đối tượng PCA từ thư viện `sklearn.decomposition` được khởi tạo với `n_components=PCA_COMPONENTS` (đặt bằng 2) để giảm dữ liệu xuống 2 chiều.
- Mô hình PCA được huấn luyện (fit) và áp dụng (transform) trên `features_scaled_df`.
- Kết quả là một mảng NumPy chứa tọa độ của mỗi cầu thủ trên 2 thành phần chính mới (PC1 và PC2).
- Tỷ lệ phương sai của dữ liệu gốc được giải thích bởi 2 thành phần chính này được tính toán và in ra màn hình.

Bước 3: Trực quan hóa Kết quả phân cụm 2D:

- Hai thành phần chính (PC1, PC2) được sử dụng làm trục X và Y cho biểu đồ.
- Sử dụng thư viện `seaborn` và `matplotlib`, một biểu đồ phân tán (scatter plot) được vẽ, trong đó mỗi điểm đại diện cho một cầu thủ.
- Các điểm được tô màu dựa trên cột `Cluster` từ `df_clustered_sorted`, giúp phân biệt các nhóm cầu thủ khác nhau.
- Biểu đồ được đặt tiêu đề, nhãn trục, chú giải (legend) và lưới (grid) để dễ đọc.
- Biểu đồ trực quan hóa này được lưu vào file ảnh có tên định dạng `kmeans_pca_{optimal_k}_clusters_viz.png` trong thư mục `pca_kmeans_viz_results`.

Bước 4: Lưu dữ liệu cuối cùng:

- Các tọa độ PC1, PC2 được ghép vào DataFrame `df_clustered_sorted` để tạo thành `df_pca_clustered`.
- DataFrame này, chứa thông tin cầu thủ, nhãn cụm K-Means, và hai thành phần chính PCA, được lưu vào file CSV có tên định dạng `player_clusters_with_pca_{optimal_k}.csv` trong cùng thư mục kết quả.

3.2.3 Kết quả và nhận xét

Hình 3.3: Phân cụm cầu thủ K-Means (K=6) - Trực quan hóa PCA 2D.

Nhận xét về biểu đồ PCA: Tỷ lệ phương sai được giải thích bởi 2 thành phần chính (PC1 và PC2): 0.6499 (tức là 64.99%). Điều này cho thấy hai thành phần chính đầu tiên giải thích được khoảng 65% tổng phương sai trong dữ liệu, còn lại 35% thông tin nằm ở các thành phần khác. Biểu đồ cho thấy các cụm được phân chia khá rõ ràng, nhưng có một số vùng chồng lấn giữa các cụm, đặc biệt ở khu vực trung tâm (gần giao điểm PC1 = 0, PC2 = 0). Điều này cho thấy một số cầu thủ có đặc điểm tương đối giống nhau giữa các cụm, có thể do các thông số thống kê của họ (ví dụ: số bàn thắng, kiến tạo, hoặc các chỉ số phòng ngự) không khác biệt lớn.

- Cụm 0 (màu vàng) tập trung chủ yếu ở góc trái trên của biểu đồ (PC1 âm, PC2 dương), có thể đại diện cho một nhóm cầu thủ với các đặc điểm riêng biệt (ví dụ: các hậu vệ hoặc thủ môn, dựa trên dữ liệu CSV, vì họ có xu hướng không ghi bàn và tham gia nhiều vào phòng ngự).

- Cụm 1 (màu xanh lá nhạt) nằm ở phía dưới bên phải (PC1 dương, PC2 âm), có thể là nhóm các tiền đạo hoặc cầu thủ tấn công ghi bàn nhiều (như Alexander Isak, Yoane Wissa trong cụm này).
- Cụm 2 (màu xanh lam) phân bố rộng ở khu vực trung tâm và bên phải, có thể là các tiền vệ hoặc hậu vệ cánh với vai trò cân bằng giữa tấn công và phòng ngự (như Alexis Mac Allister, Andrew Robertson).
- Cụm 3 (màu xanh lam nhạt) tập trung ở góc trái dưới (PC1 âm, PC2 âm), có thể là các cầu thủ trẻ hoặc dự bị với ít thời gian thi đấu và ít đóng góp vào các chỉ số tấn công (như Abdul Fatawu Issahaku, Adam Lallana).
- Cụm 4 (màu tím) nằm ở phía bên phải (PC1 dương), có thể là các cầu thủ tấn công hoặc cánh với nhiều pha đi bóng và tham gia tấn công (như Alejandro Garnacho, Adama Traoré).
- Cụm 5 (màu hồng) phân bố rải rác nhưng tập trung ở khu vực PC2 dương lớn, có thể là các trung vệ hoặc cầu thủ phòng ngự với chỉ số phòng ngự cao (như Virgil van Dijk, Wout Faes).

3.2.4 Kết luận

Script `pca_visual_kmeans.py` thực hiện thành công yêu cầu trực quan hóa kết quả phân cụm K-Means của cầu thủ bằng cách sử dụng PCA để giảm chiều dữ liệu xuống 2D. Kết quả là một biểu đồ phân tán 2D và một bộ dữ liệu mới kèm theo tọa độ PCA, cho phép đánh giá trực quan ban đầu về sự phân tách của các cụm. Mặc dù phương pháp này có những hạn chế nhất định liên quan đến mất mát thông tin, nó vẫn là một công cụ hữu ích trong bộ công cụ phân tích dữ liệu để khám phá và trình bày kết quả phân cụm một cách dễ hiểu hơn.

Chương 4

Thu thập giá trị chuyển nhượng và xây dựng mô hình ước tính giá trị cầu thủ

4.1 Thu thập và xử lý giá trị chuyển nhượng cầu thủ mùa giải 2024-2025

4.1.1 Giới thiệu

Phần báo cáo này trình bày quy trình thu thập dữ liệu giá trị chuyển nhượng của các cầu thủ bóng đá cho mùa giải 2024-2025 từ trang web `https://www.footballtransfers.com`. Quá trình này tập trung vào việc lấy thông tin của những cầu thủ có thời gian thi đấu thực tế lớn hơn 900 phút, dựa trên dữ liệu được cung cấp. File mã nguồn chính là `transfer_values_2024-2025.py`.

4.1.2 Cấu trúc thư mục và thư viện sử dụng

- Cấu trúc thư mục: Tổ chức gồm file code chính `transfer_values_2024-2025.py` và file cấu hình `config.json` trong thư mục gốc.
- Dữ liệu đầu vào được lấy từ thư mục `../problem_1/`, và kết quả được lưu vào thư mục `OUTPUT/` (bao gồm `raw_data.csv`, `player_transfer_values.csv`, `estimation_data.csv`).
- Thư viện chính sử dụng: Python, `os`, `re`, `time`, `pandas`, `json`, `BeautifulSoup` (từ `bs4`), và `selenium` (cùng các modules con của nó).

4.1.3 Quy trình thực hiện

Bước 1: Tải cấu hình

Script bắt đầu bằng việc tải các thông số cấu hình từ file `config.json` thông qua hàm `load_config`. File này chứa các đường dẫn quan trọng (ví dụ: `paths.part1_results_relative` cho file dữ liệu thời gian thi đấu, `paths.output_folder` cho thư mục lưu kết quả), URL mục tiêu (`scraping.transfer_url`), thông tin user-agent (`scraping.user_agent` để mô phỏng trình duyệt), các CSS selector (trong `selectors`) để xác định vị trí các

mẫu dữ liệu cụ thể trên trang web (như tên cầu thủ, ETV, nút "Next page"), và ngưỡng thời gian thi đấu tối thiểu (`processing.min_minutes_threshold`).

Bước 2: Thu thập dữ liệu (Web Scraping) - thực hiện trong hàm `scrape_transfer_data`:

- Hàm `scrape_transfer_data` sử dụng Selenium để mở URL của trang danh sách cầu thủ (ví dụ: Premier League) được chỉ định trong `config.json`.
- Script duyệt qua từng trang trong danh sách cầu thủ.
- Selenium đợi cho các yếu tố cần thiết của trang, như bảng dữ liệu cầu thủ (`selectors.player_table`), được tải hoàn tất trước khi tiến hành trích xuất.
- Đối với mỗi cầu thủ (mỗi hàng `selectors.player_row` trong bảng), các thông tin sau được trích xuất dựa trên các CSS selector định nghĩa trong `config.json`:
  - Tên cầu thủ (`selectors.player_name`)
  - Tên đội (`selectors.team_name`)
  - Giá trị chuyển nhượng ước tính (ETV - `selectors.etv`)
  - Tuổi (`selectors.age`)
  - Vị trí (`selectors.position`)
  - Kỹ năng (`selectors.skill`)
  - Tiềm năng (`selectors.potential`)
- Xử lý giá trị chuyển nhượng (ETV) ngay khi thu thập: Dữ liệu ETV thô (ví dụ: "€50.5m") được làm sạch và chuyển đổi sang đơn vị triệu Euro ngay tại bước này bằng hàm `clean_transfer_value`. Hàm này:
  - Loại bỏ các ký tự tiền tệ (ví dụ: '€', '$') và dấu phẩy.
  - Nhận diện và xử lý các hậu tố 'm' (triệu) hoặc 'k' (nghìn) để quy đổi về giá trị số đầy đủ.
  - Chuẩn hóa giá trị về đơn vị triệu (ví dụ 50.5).
  - Nếu giá trị không hợp lệ, nó sẽ trả về `None`.
- Tương tự, các giá trị số khác như Tuổi, Kỹ năng, Tiềm năng được trích xuất dưới dạng text và hàm `safe_get_numeric` cố gắng chuyển đổi chúng thành dạng số, xử lý dấu phẩy nếu có.
- Cơ chế "Chuyển nút"(Phân trang):
  - Sau khi xử lý hết cầu thủ trên một trang, script tìm nút "Next page".
  - Hàm `get_next_page_button` được gọi, sử dụng `WebDriverWait` để chờ tối đa 5 giây cho đến khi nút "Next page"(được xác định bởi selector `selectors.next_page_enabled` từ `config.json`, ví dụ: `button.pagination_next_button:not([disabled])`) trở nên khả dụng và có thể click được. Selector này đảm bảo chỉ chọn nút đang hoạt động.
  - Nếu không tìm thấy nút "Next page"hợp lệ (ví dụ, ở trang cuối cùng hoặc có lỗi tải trang), `get_next_page_button` trả về `None`, và vòng lặp thu thập dữ liệu trên các trang sẽ kết thúc.
  - Nếu nút được tìm thấy:
    - Script thực hiện `driver.execute_script("arguments[0].scrollIntoView({block: 'center'});", next_btn)` để cuộn nút vào giữa tầm nhìn của trình

duyệt. Việc này giảm thiểu nguy cơ nút bị che bởi các phần tử khác và không thể click được.

    * Một khoảng dừng ngắn ngẫu nhiên (`time.sleep(random.uniform(0.4, 0.8))`) được áp dụng.
    * Script thử click nút bằng `next_btn.click()`. Nếu xảy ra `ElementClickInterceptedException` (nghĩa là một phần tử khác đang che nút), script sẽ chuyển sang phương án dự phòng: thực hiện click bằng JavaScript với `driver.execute_script("arguments[0].click();", next_btn)`.
    * Sau khi click thành công, một khoảng dừng dài hơn (`time.sleep(random.uniform(2.0, 3.5))`) được áp dụng để chờ trang mới tải hoàn tất trước khi tiếp tục vòng lặp.

- Sau khi hàm `scrape_transfer_data` hoàn tất việc duyệt qua tất cả các trang và thu thập thông tin của các cầu thủ vào một cấu trúc dữ liệu tạm thời (một dictionary trong Python), toàn bộ khối dữ liệu này (`scraped_data`) sẽ được chuyển sang bước tiếp theo để xử lý, nơi nó sẽ được chuyển đổi thành DataFrame và lưu dưới dạng file dữ liệu thô (`raw_data.csv`).

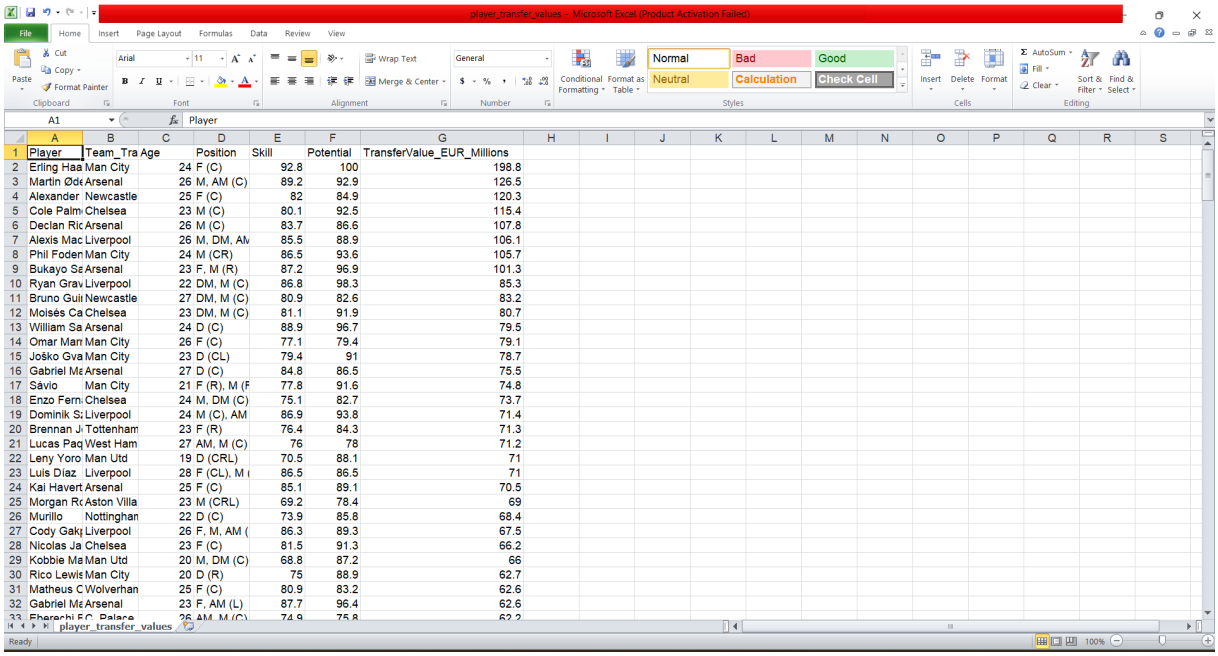
Bước 3: Lọc dữ liệu dựa trên thời gian thi đấu: Đây là bước cốt lõi để đảm bảo chỉ những cầu thủ đáp ứng tiêu chí về thời gian thi đấu được giữ lại.

- **Lấy danh sách cầu thủ hợp lệ (`get_valid_players_from_part1`):** Hàm này đọc file CSV bên ngoài `results1.csv` (đường dẫn được khai báo trong `config.json` qua key `paths.part1_results_relative`).
- Từ file `results1.csv` này, script sẽ sử dụng chủ yếu hai cột quan trọng để lọc cầu thủ:
  - Cột chứa tên cầu thủ, được xác định bởi giá trị của key `processing.part1_player_column` trong `config.json` (ví dụ: "Name").
  - Cột chứa thời gian thi đấu của cầu thủ, được xác định bởi giá trị của key `processing.part1_minutes_column` trong `config.json` (ví dụ: "Playing Time: minutes").
- Sau đó, thời gian thi đấu từ cột này sẽ được chuyển đổi sang dạng số và so sánh với ngưỡng `min_minutes_threshold` (900 phút) để tạo ra danh sách các tên cầu thủ "hợp lệ".

Bước 4: Áp dụng bộ lọc vào dữ liệu đã thu thập

- Trong hàm `process_transfer_data`, nếu danh sách cầu thủ hợp lệ (`valid_player_set`) không rỗng, script sẽ tiến hành lọc.
- Một cột tạm thời `Player_Normalized` được tạo trong `df_raw` bằng cách chuẩn hóa cột 'Player' (loại bỏ khoảng trắng thừa).
- `df_raw` sau đó được lọc để chỉ giữ lại những hàng mà giá trị trong cột `Player_Normalized` có trong `valid_player_set`.
- Cột tạm thời `Player_Normalized` được xóa sau khi lọc.
- Nếu không có danh sách cầu thủ hợp lệ nào được tìm thấy từ file CSV đầu vào (ví dụ: file không tồn tại, không có cầu thủ nào thỏa mãn ngưỡng), thì `df_raw` (dữ liệu chưa lọc theo thời gian thi đấu) sẽ được trả về.

4.1.4 Kết quả



Player	Team_Tra	Age	Position	Skill	Potential	TransferValue_EUR_Millions
Erling Haa	Man City	24	F (C)	92.8	100	198.8
Martin Ød	Arsenal	26	M, AM (C)	89.2	92.9	126.5
Alexander	Newcastle	25	F (C)	82	84.9	120.3
Cole Palm	Chelsea	23	M (C)	80.1	92.5	115.4
Declan Ric	Arsenal	26	M (C)	83.7	86.6	107.8
Alexis Mac	Liverpool	26	M, DM, AM	85.5	88.9	106.1
Phil Foden	Man City	24	M (CR)	86.5	93.6	105.7
Bukayo Sa	Arsenal	23	F, M (R)	87.2	96.9	101.3
Ryan Grav	Liverpool	22	DM, M (C)	86.8	98.3	85.3
Bruno Guii	Newcastle	27	DM, M (C)	80.9	82.6	83.2
Moisés Ca	Chelsea	23	DM, M (C)	81.1	91.9	80.7
William Sa	Arsenal	24	D (C)	88.9	96.7	79.5
Omar Marı	Man City	26	F (C)	77.1	79.4	79.1
Joško Gva	Man City	23	D (CL)	79.4	91	78.7
Gabriel Ma	Arsenal	27	D (C)	84.8	86.5	75.5
Sávio	Man City	21	F (R), M (F	77.8	91.6	74.8
Enzo Fern	Chelsea	24	M, DM (C)	75.1	82.7	73.7
Dominik Sz	Liverpool	24	M (C), AM	86.9	93.8	71.4
Brennan Jo	Tottenham	23	F (R)	76.4	84.3	71.3
Lucas Paq	West Ham	27	AM, M (C)	76	78	71.2
Leny Yoro	Man Utd	19	D (CRL)	70.5	88.1	71
Luis Díaz	Liverpool	28	F (CL), M (	86.5	86.5	71
Kai Havert	Arsenal	25	F (C)	85.1	89.1	70.5
Morgan Ro	Aston Villa	23	M (CRL)	69.2	78.4	69
Murillo	Nottinghan	22	D (C)	73.9	85.8	68.4
Cody Gakp	Liverpool	26	F, M, AM (	86.3	89.3	67.5
Nicolas Ja	Chelsea	23	F (C)	81.5	91.3	66.2
Kobbie Ma	Man Utd	20	M, DM (C)	68.8	87.2	66
Rico Lewis	Man City	20	D (R)	75	88.9	62.7
Matheus C	Wolverhan	25	F (C)	80.9	83.2	62.6
Gabriel Ma	Arsenal	23	F, AM (L)	87.7	96.4	62.6

Hình 4.1: Ví dụ dữ liệu giá trị chuyển nhượng cầu thủ sau khi xử lý.

Quy trình tự động hóa thu thập và xử lý dữ liệu đã được triển khai thành công thông qua script `transfer_values_2024-2025.py` và file cấu hình `config.json`. Kết quả của quá trình này là đã thu thập và lọc được thông tin chi tiết của **284 cầu thủ** có thời gian thi đấu thực tế trên 900 phút trong mùa giải 2024-2025 từ trang `footballtransfers.com`. Dữ liệu thu được cho mỗi cầu thủ bao gồm các chỉ số quan trọng sau:

- Đội (Team)
- Tuổi (Age)
- Vị trí (Position)
- Kỹ năng (Skill)
- Tiềm năng (Potential)
- Giá trị chuyển nhượng ước tính (TransferValue_EUR_Millions)

Toàn bộ dữ liệu này đã được lưu trữ trong các file output thông qua hàm `save_data`:

- `player_transfer_values.csv`: dữ liệu cuối cùng về giá trị chuyển nhượng của cầu thủ đã được xử lý và lọc
- `estimation_data.csv`: chuẩn bị để sử dụng trong phần sau: dự đoán giá trị cầu thủ

4.2 Xây dựng mô hình ước tính giá trị cầu thủ bóng đá

4.2.1 Giới thiệu

Thị trường chuyển nhượng cầu thủ bóng đá là một lĩnh vực phức tạp và năng động, nơi việc định giá chính xác cầu thủ đóng vai trò quan trọng đối với các câu lạc bộ. Phần báo cáo này trình bày phương pháp xây dựng một mô hình học máy để ước tính giá trị chuyển nhượng của cầu thủ, dựa trên các số liệu thống kê về hiệu suất, tiềm năng và các yếu tố khác. Mục tiêu là đề xuất một quy trình có hệ thống từ thu thập dữ liệu, kỹ thuật đặc trưng, lựa chọn mô hình đến đánh giá và giải thích kết quả. Biến mục tiêu là giá trị chuyển nhượng của cầu thủ, được biến đổi bằng hàm logarit (`Log_TransferValue`) để ổn định phương sai và giảm tác động của các giá trị ngoại lai.

4.2.2 Mô tả dữ liệu và tiền xử lý

Nguồn dữ liệu và kết hợp

- **Nguồn dữ liệu:**
  - Dữ liệu ước tính (`estimation_data.csv`): Chứa các thông tin cơ bản và một số chỉ số ban đầu của cầu thủ.
  - Dữ liệu kết quả (`results1.csv`): Có thể chứa các chỉ số hiệu suất chi tiết hơn từ một nguồn khác.
- **Kết hợp dữ liệu:**
  - Hai bộ dữ liệu được kết hợp (merged) dựa trên thông tin của cầu thủ và đội bóng.
  - Tên đội bóng đã được chuẩn hóa để đảm bảo việc kết hợp chính xác (ví dụ: "B'mouth"thành "Bournemouth").
- **Kết quả sau khi kết hợp:** (283, 85), nghĩa là có 283 mẫu (cầu thủ) và 85 đặc trưng ban đầu. Điều này được thực hiện trong hàm `load_and_merge_data`.

Kỹ thuật đặc trưng (Feature Engineering)

Mục tiêu của kỹ thuật đặc trưng là tạo ra các biến mới có ý nghĩa hơn từ dữ liệu thô, giúp mô hình học tốt hơn. Các bước chính bao gồm:

- **Tạo vị trí chính (PrimaryPosition):** Từ cột `Position_x` (có thể chứa nhiều vị trí), chỉ lấy vị trí đầu tiên làm vị trí chính của cầu thủ. Các giá trị thiếu hoặc không xác định được gán là 'Unknown'.
- **Chuẩn hóa và chuyển đổi kiểu dữ liệu số:** Các cột như 'Skill', 'Potential', 'Age_x', 'Expected: expected goals (xG)', 'Expected: expected Assist Goals (xAG)', 'TransferValue_EUR_Millions' được chuyển đổi sang kiểu số. Các lỗi chuyển đổi (nếu có) được xử lý bằng cách gán giá trị `NaN`.
- **Tính toán các chỉ số hiệu suất trên 90 phút:**
  - `GoalContrib_per90`: Tổng số bàn thắng và kiến tạo trên 90 phút.

- `DefensiveActions_per90`: Tổng số pha tắc bóng thành công (TklW) và chặn bóng (Int) trên 90 phút.
- `Progression_per90`: Tổng số pha dẫn bóng và chuyền bóng tịnh tiến trên 90 phút.

- **Chuẩn hóa các cột tỷ lệ phần trăm:** Các cột như 'Total: Pass completion (Cmp%)', 'Aerial Duels: Won%' được chuyển từ dạng chuỗi (ví dụ: "85%") sang dạng số thập phân (ví dụ: 0.85).

- **Biến đổi biến mục tiêu:**
  - Giá trị chuyển nhượng ('TransferValue_EUR_Millions') được đảm bảo không âm (các giá trị âm được cắt về 0).
  - Tạo cột `Log_TransferValue` bằng cách áp dụng hàm `np.log1p` (logarit tự nhiên của 1+x) cho `TransferValue_EUR_Millions`. Phép biến đổi này giúp:
    * Giảm độ xiên của phân phối giá trị chuyển nhượng.
    * Ổn định phương sai.
    * Đảm bảo đầu vào của hàm logarit luôn dương, ngay cả khi giá trị chuyển nhượng là 0.

- **Kết quả sau kỹ thuật đặc trưng:** (283, 90), cho thấy có 5 đặc trưng mới được tạo ra hoặc các cột được xử lý. Quá trình này được thực hiện trong hàm `advanced_feature_engineering`.

4.2.3 Lựa chọn đặc trưng và chuẩn bị dữ liệu cho mô hình

Lựa chọn Đặc trưng (Feature Selection)

Không phải tất cả các đặc trưng được tạo ra đều hữu ích cho mô hình. Các đặc trưng sau đã được lựa chọn để đưa vào mô hình: 'Skill', 'Potential', 'Age_x', 'GoalContrib_per90', 'DefensiveActions_per90', 'Progression_per90', 'Expected: expected goals (xG)', 'Expected: expected Assist Goals (xAG)', 'Total: Pass completion (Cmp%)', 'Aerial Duels: Won%', 'PrimaryPosition_enc'.

Lý do lựa chọn (suy đoán dựa trên tên đặc trưng và kiến thức chung):

- *Thuộc tính cá nhân:* 'Age_x', 'Skill', và 'Potential' là các chỉ số nền tảng đánh giá chất lượng hiện tại, kinh nghiệm và tiềm năng phát triển tương lai của cầu thủ. Tuổi tác là một yếu tố quan trọng, thường có mối quan hệ phi tuyến với giá trị cầu thủ.

- *Hiệu suất tấn công:* 'GoalContrib_per90', 'Expected: expected goals (xG)', 'Expected: expected Assist Goals (xAG)' đo lường khả năng đóng góp trực tiếp vào mặt trận tấn công.

- *Hiệu suất phòng ngự:* 'DefensiveActions_per90' phản ánh đóng góp ở mặt trận phòng ngự.

- *Khả năng phát triển bóng và kiểm soát:* 'Progression_per90' và 'Total: Pass completion (Cmp%)' cho thấy khả năng xử lý và phân phối bóng.

- *Đặc điểm thể chất/kỹ năng cụ thể:* 'Aerial Duels: Won%' quan trọng đối với một số vị trí.

- *Yếu tố vị trí:* 'PrimaryPosition' là yếu tố quan trọng vì giá trị cầu thủ thường khác nhau đáng kể giữa các vị trí.

Chuẩn bị dữ liệu

Các bước chuẩn bị dữ liệu cuối cùng trước khi đưa vào mô hình, được thực hiện trong hàm `prepare_model_data`:

- **Xử lý giá trị thiếu (Missing Values):** Đối với các đặc trưng số được chọn, giá trị thiếu được điền bằng giá trị trung bình của cột đó.
- **Chuẩn hóa đặc trưng số (Numerical Feature Scaling):** Các đặc trưng số được chuẩn hóa bằng `StandardScaler` (trừ các giá trị về trung bình 0 và phương sai 1). Điều này giúp các thuật toán nhạy cảm với thang đo (như XGBoost khi không dùng `tree_method='hist'` hoặc các mô hình dựa trên khoảng cách) hoạt động hiệu quả hơn.
- **Mã hóa đặc trưng danh mục (Categorical Feature Encoding):** Đặc trưng `PrimaryPosition` được mã hóa bằng `LabelEncoder`. Mỗi vị trí duy nhất sẽ được gán một giá trị số nguyên.
- **Tạo ma trận đặc trưng (X) và vector mục tiêu (y):**
  - X: Ma trận chứa các đặc trưng đã được xử lý (chuẩn hóa và mã hóa). Kích thước: (283, 10).
  - y: Vector chứa biến mục tiêu `Log_TransferValue`. Kích thước: (283,).
- **Kiểm tra dữ liệu:** Đảm bảo ma trận X không rỗng và số lượng mẫu trong X khớp với y.

4.2.4 Lựa chọn mô hình và huấn luyện

Lựa chọn Mô hình

Mô hình được lựa chọn để ước tính giá trị cầu thủ là XGBoost (Extreme Gradient Boosting). **Lý do lựa chọn mô hình XGBoost để ước tính giá trị cầu thủ:**

- *Khả năng nắm bắt mối quan hệ phức tạp và phi tuyến:* Giá trị cầu thủ được quyết định bởi sự tương tác đa chiều giữa nhiều yếu tố như tuổi tác, kỹ năng, tiềm năng và hiệu suất thi đấu. XGBoost, với cấu trúc dựa trên tập hợp các cây quyết định, xuất sắc trong việc tự động phát hiện và mô hình hóa các mối quan hệ phức tạp này (ví dụ: ảnh hưởng của tuổi không phải lúc nào cũng tuyến tính, hoặc tiềm năng có giá trị khác nhau ở các độ tuổi khác nhau) mà không đòi hỏi phải định nghĩa trước. Điều này cực kỳ quan trọng để phản ánh đúng bản chất của việc định giá cầu thủ.
- *Hiệu suất dự đoán vượt trội và kiểm soát overfitting hiệu quả:* XGBoost nổi tiếng với khả năng đạt được độ chính xác cao trên dữ liệu dạng bảng. Thuật toán này tích hợp các cơ chế điều chuẩn (regularization L1, L2) và kỹ thuật tối ưu hóa (như `early_stopping_rounds` được sử dụng) giúp ngăn chặn mô hình trở nên quá phức tạp hoặc "học vẹt"dữ liệu huấn luyện. Kết quả là mô hình có khả năng tổng quát hóa tốt hơn trên dữ liệu mới, đảm bảo tính tin cậy của các dự đoán giá trị cầu thủ.

- *Tối ưu hóa hiệu quả và cung cấp diễn giải về đặc trưng:* XGBoost được thiết kế để huấn luyện nhanh chóng, ngay cả trên các bộ dữ liệu tương đối lớn, nhờ vào các kỹ thuật như xử lý song song và phương pháp tạo histogram (`tree_method='hist'`). Quan trọng hơn, nó cung cấp thông tin chi tiết về mức độ quan trọng của từng đặc trưng (ví dụ: `Age_x`, `Skill`, `GoalContrib_per90`), cho phép chúng ta hiểu rõ yếu tố nào có ảnh hưởng lớn nhất đến việc định giá. Điều này không chỉ giúp đánh giá mô hình mà còn mang lại những hiểu biết giá trị cho các nhà phân tích và quản lý bóng đá.

Huấn luyện và Tối ưu hóa Tham số

Quá trình huấn luyện được thực hiện trong hàm `train_model`:

- **Phân chia dữ liệu:** Dữ liệu (X, y) được chia thành tập huấn luyện (train set) và tập kiểm thử (test set) với tỷ lệ 80:20 (`test_size=0.2`). `X_train` có kích thước (226,11).
- **Tối ưu hóa siêu tham số (Hyperparameter Tuning):**
  - Sử dụng `RandomizedSearchCV` để tìm bộ siêu tham số tốt nhất cho XGBoost.
  - `RandomizedSearchCV` thử nghiệm một số lượng tổ hợp tham số ngẫu nhiên (`n_iter=20`) từ một không gian tìm kiếm được định nghĩa trước (`param_grid`).
  - Đánh giá chéo (Cross-validation) với `cv=5` được sử dụng trong quá trình tìm kiếm để đảm bảo tính ổn định của kết quả.
  - Chỉ số đánh giá được sử dụng trong quá trình tìm kiếm là `r2` (R-squared).
- **Siêu tham số tốt nhất được tìm thấy:**
  - subsample: 0.6
  - n_estimators: 1000
  - max_depth: 6
  - learning_rate: 0.05
  - gamma: 0.5
  - colsample_bytree: 0.6
- **Huấn luyện mô hình cuối cùng:** Mô hình XGBoost được huấn luyện lại trên toàn bộ tập huấn luyện (`X_train`, `y_train`) với bộ siêu tham số tốt nhất vừa tìm được. Có sử dụng `early_stopping_rounds=50` và `eval_set=[(X_test, y_test)]` để theo dõi hiệu suất trên tập kiểm thử và dừng sớm nếu không có cải thiện, nhằm tránh overfitting.
- **Môi trường huấn luyện:** Quá trình huấn luyện mô hình XGBoost được thực hiện trên CPU.

4.2.5 Đánh giá mô hình

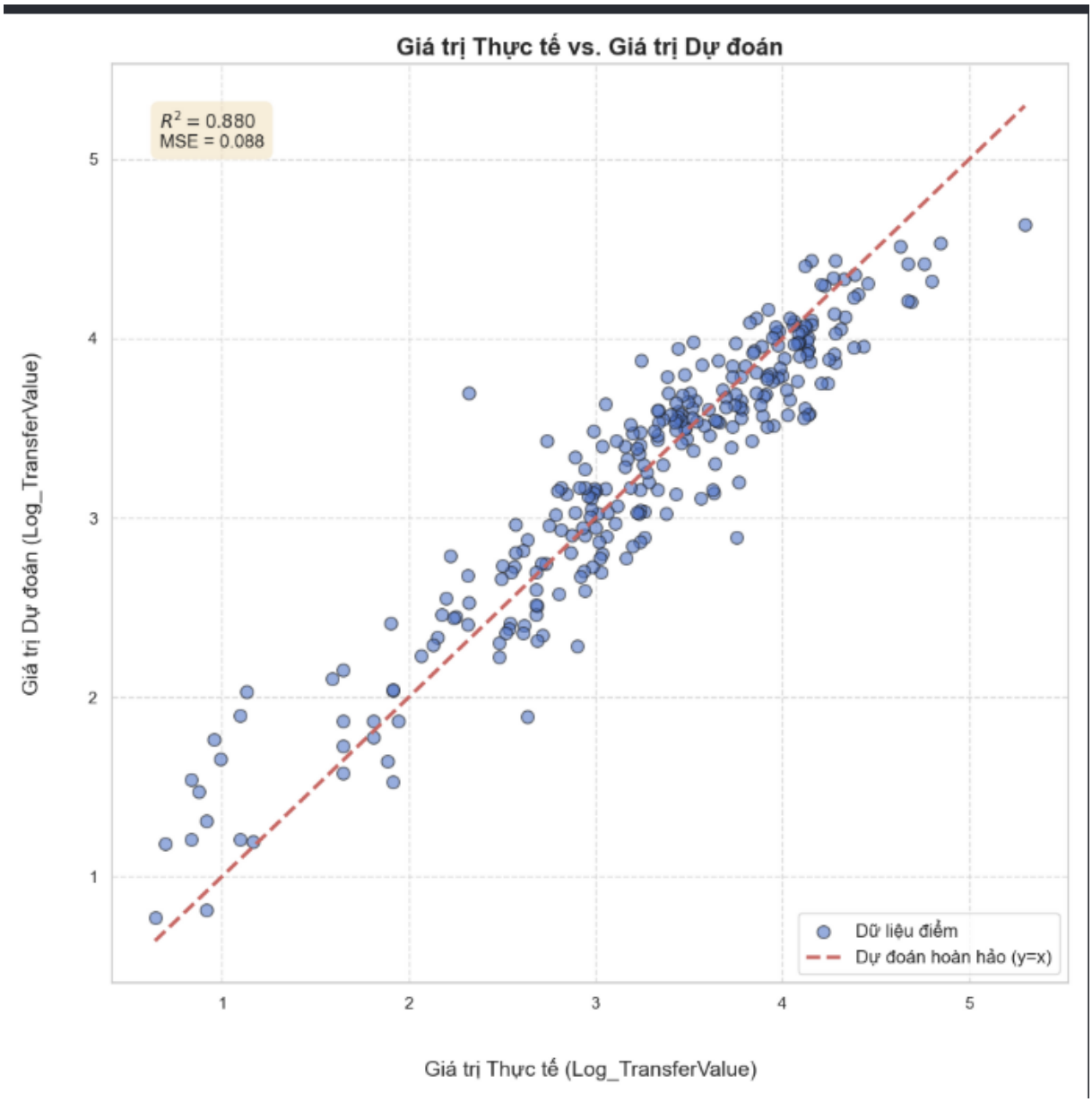
Mô hình được đánh giá trên toàn bộ dữ liệu đã qua xử lý (X, y).

- **R^2 score (Hệ số xác định R bình phương = 0.880):** Điều này có nghĩa là khoảng 88.0% sự biến thiên của `Log_TransferValue` có thể được giải thích bởi các đặc trưng đầu vào của mô hình. Đây là một kết quả khá tốt, cho thấy mô hình có khả năng nắm bắt được các yếu tố quan trọng ảnh hưởng đến giá trị cầu thủ.
- **MSE (Mean Squared Error - Sai số toàn phương trung bình = 0.088):** Đây là giá trị trung bình của bình phương sai số giữa giá trị dự đoán và giá trị thực tế (trên thang đo logarit). Giá trị MSE càng nhỏ càng tốt.

4.2.6 Phân tích và trực quan hóa kết quả

Các biểu đồ trực quan hóa giúp hiểu rõ hơn về hiệu suất và hoạt động bên trong của mô hình.

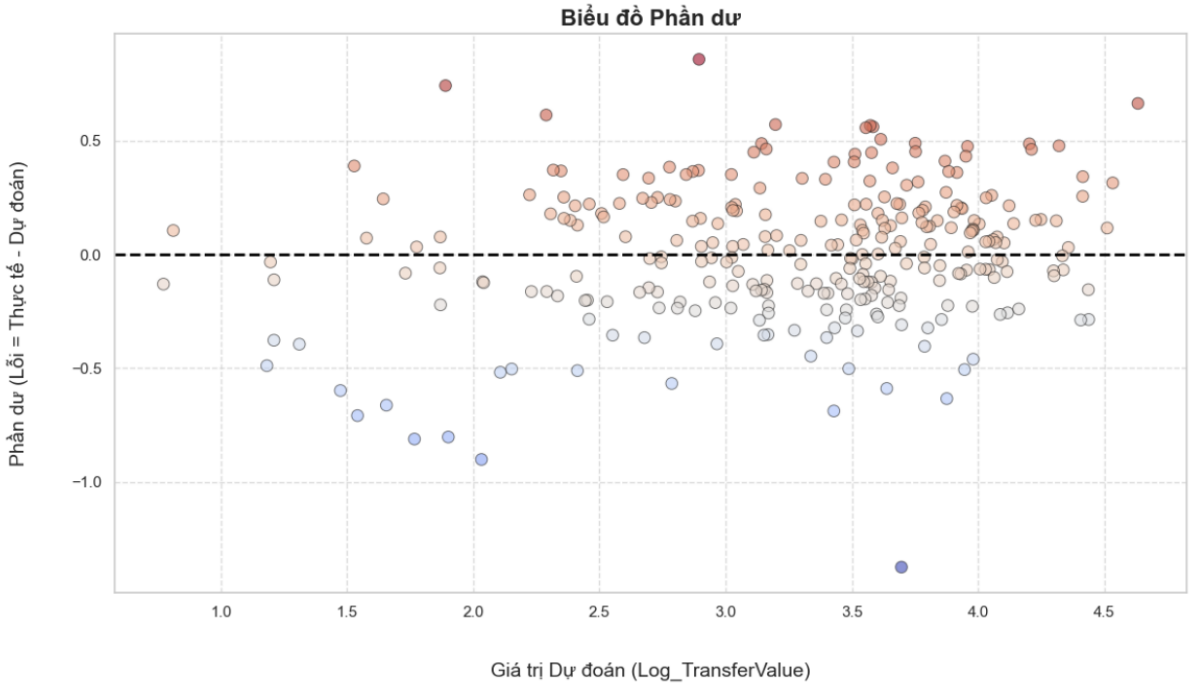
Biểu đồ giá trị thực tế vs. giá trị dự đoán (`actual_vs_predicted.png`)



Hình 4.2: Biểu đồ giá trị thực tế vs giá trị dự đoán.

Dựa vào biểu đồ: Mô hình dự đoán giá trị cầu thủ **khá tốt**, thể hiện qua các điểm dữ liệu tập trung tương đối sát đường chéo (dự đoán hoàn hảo) và chỉ số R^2 cao (0.880). Tuy nhiên, mô hình **vẫn có sai số** (MSE = 0.088, trên thang log), với một số điểm dự đoán lệch đáng kể so với giá trị thực tế, đặc biệt có thể ở các khoảng giá trị thấp và cao.

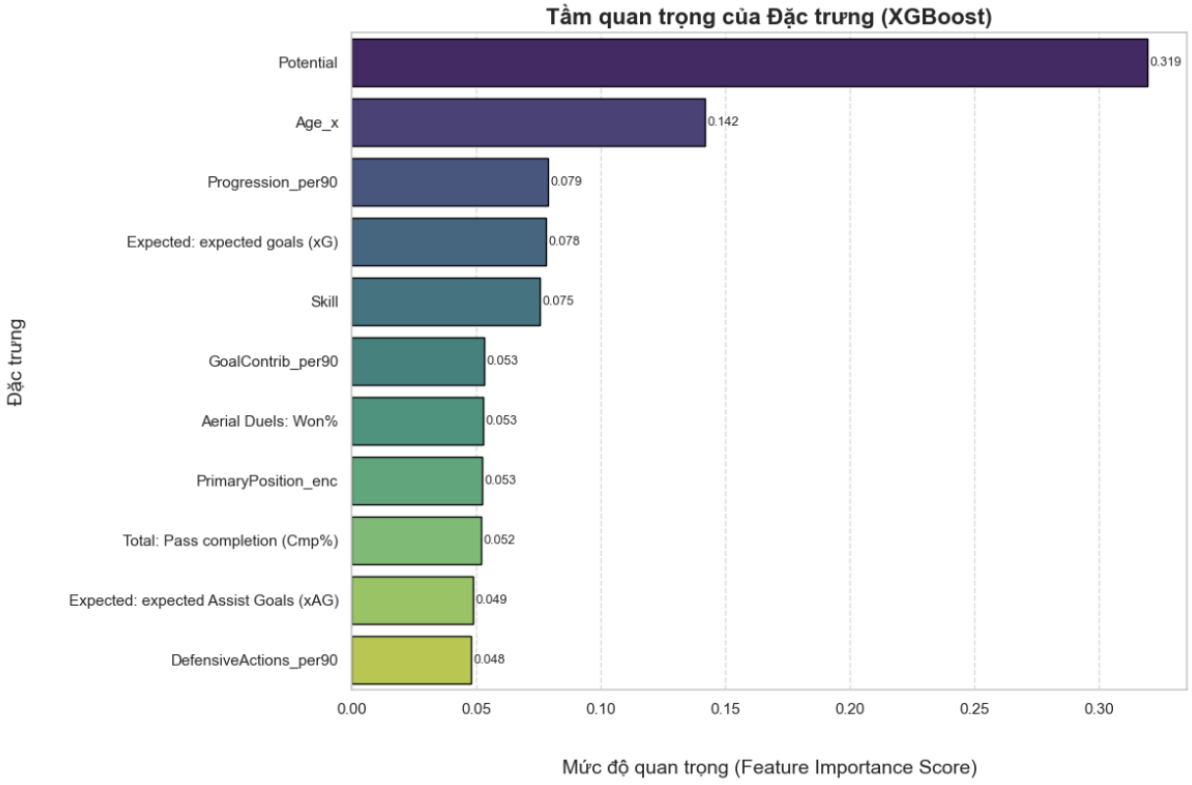
Biểu đồ Phần dư (`residual_plot.png`)



Hình 4.3: Biểu đồ Phần dư.

Biểu đồ phần dư cho thấy phần lớn các điểm phân bố gần đường 0 trên trục Phần dư, với một số ít điểm lệch đáng kể, đặc biệt ở các giá trị dự đoán cao (`Log_TransferValue` lớn). Điều này cho thấy mô hình XGBoost có hiệu quả tốt trong việc dự đoán giá trị cầu thủ, với độ chính xác cao ở phần lớn dữ liệu, nhưng có thể cần cải thiện đối với các trường hợp ngoại lệ hoặc giá trị cao.

Tầm quan trọng Đặc trưng của XGBoost (`xgb_feature_importance.png`)



Hình 4.4: Biểu đồ Tầm quan trọng của Đặc trưng (XGBoost).

Biểu đồ Tầm quan trọng Đặc trưng của XGBoost cho thấy "Potential"là đặc trưng quan trọng nhất với điểm số 0.319, theo sau là "Age"(0.142). Các đặc trưng như "Progression_per90"(0.079), "Expected: expected goals (xG)"(0.078), và "Skill"(0.075) cũng có ảnh hưởng đáng kể. Các đặc trưng khác như "DefensiveActions_per90"(0.048) có mức độ quan trọng thấp hơn. Điều này cho thấy mô hình phụ thuộc mạnh vào tiềm năng và tuổi của cầu thủ, đồng thời cân nhắc các chỉ số hiệu suất như tiến trình và kỹ năng.

4.2.7 Kết luận và hướng phát triển

Kết luận

Phương pháp đề xuất đã xây dựng thành công một mô hình XGBoost để ước tính giá trị cầu thủ (biến đổi logarit) với hiệu suất khá tốt (R2=0.880). Quá trình này bao gồm các bước quan trọng từ kết hợp dữ liệu, kỹ thuật đặc trưng, lựa chọn và chuẩn bị đặc trưng, đến huấn luyện, tối ưu hóa và đánh giá mô hình. Các đặc trưng được lựa chọn như `Skill`, `Potential`, `Age` các chỉ số hiệu suất tấn công/phòng ngự trên 90 phút, và thông số về chuyền bóng/không chiến đã cho thấy khả năng đóng góp vào việc dự đoán giá trị. Việc sử dụng `RandomizedSearchCV` giúp tìm ra bộ siêu tham số tốt cho XGBoost. Các trực quan hóa cung cấp khả năng giải thích sâu sắc về hoạt động của mô hình.

Hướng phát triển

- **Đánh giá trên tập kiểm thử độc lập:** Để có đánh giá khách quan hơn về khả năng tổng quát hóa, cần đánh giá mô hình trên một tập dữ liệu kiểm thử hoàn toàn riêng biệt (không được sử dụng trong quá trình tối ưu hóa tham số).
- **Thử nghiệm các mô hình khác:** So sánh XGBoost với các thuật toán khác như LightGBM, CatBoost, hoặc các mô hình mạng neural đơn giản.
- **Kỹ thuật đặc trưng nâng cao hơn:**
  - Xem xét các tương tác giữa các đặc trưng (ví dụ: `Skill` * `GoalContrib_per90`).
  - Thu thập thêm dữ liệu (ví dụ: thông tin về giải đấu, thông tin hợp đồng, thành tích đội bóng).
- **Phân tích lỗi chi tiết:** Xem xét các trường hợp mà mô hình dự đoán sai nhiều nhất để hiểu rõ hơn về những hạn chế của mô hình.
- **Xử lý vấn đề mất cân bằng dữ liệu (nếu có):** Nếu giá trị cầu thủ phân bố rất không đều, có thể cần các kỹ thuật xử lý mất cân bằng.
- **Cập nhật mô hình định kỳ:** Giá trị cầu thủ và các yếu tố ảnh hưởng thay đổi theo thời gian, do đó mô hình cần được huấn luyện lại và cập nhật định kỳ với dữ liệu mới.

Tài liệu tham khảo

[1] XGBoost Developers. (2024). *XGBoost Documentation (Release 3.0.0)*. Truy cập ngày 5 tháng 5 năm 2025, từ https://xgboost.readthedocs.io/en/release_3.0.0/

[2] Ông Xuân Hồng (2017, 21 tháng 12). *XGBoost – thuật toán giành chiến thắng tại nhiều cuộc thi Kaggle*. Truy cập ngày 5 tháng 5 năm 2025, từ https://ongxuanhong.wordpress.com/2017/12/21/xgboost-thuat-toan-gianh-chien-thang-tai-nhieu-cuoc-thi-kaggle/

[3] Li, C., Kampakis, S., & Treleaven, P. (2022). *Machine Learning Modeling to Evaluate the Value of Football Players*. arXiv preprint arXiv:2207.11361. Truy cập ngày 5 tháng 5 năm 2025, từ https://arxiv.org/pdf/2207.11361.pdf