HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VỄN THÔNG

BỘ MÔN LẬP TRÌNH NGÔN NGỮ PYTHON

Giảng viên hướng dẫn : KIM NGỌC BÁCH

Lớp : D23CQCE06-B

Họ Tên : ĐINH ĐỨC NGHĨA

MSV : B23DCCE072

Bài 1: THU THẬP DỮ LIỆU THỐNG KÊ CẦU THỦ PREMIER LEAGUE TỪ FBREF

1. Giới thiệu chung

Bài code này được viết bằng ngôn ngữ Python và sử dụng các thư viện như Selenium và BeautifulSoup để tự động thu thập dữ liệu thống kê về các cầu thủ đang thi đấu tại giải bóng đá Ngoại hạng Anh (Premier League) từ trang web FBref. Mục tiêu của code là trích xuất thông tin từ nhiều bảng thống kê khác nhau trên trang web và tổng hợp chúng thành một bộ dữ liệu duy nhất, sau đó lưu trữ vào file định dạng CSV để tiện cho việc phân tích sau này.

2. Các thư viện đã sử dụng

- **pandas:** Thư viện này rất quan trọng để tạo và quản lý DataFrame, là cấu trúc dữ liệu dạng bảng để lưu trữ dữ liệu thu thập được một cách có tổ chức.
- **selenium:** Đây là một công cụ mạnh mẽ để tự động hóa các tương tác với trình duyệt web. Trong bài code này, Selenium được sử dụng để mở trang web FBref, chờ cho dữ liệu được tải đầy đủ, và lấy mã nguồn HTML của trang. Em đã sử dụng Chrome WebDriver để thực hiện việc này ở chế độ headless (không hiển thị giao diện đồ họa) để tiết kiệm tài nguyên.
- **webdriver_manager:** Thư viện này giúp tự động quản lý driver cho trình duyệt Chrome, đảm bảo rằng phiên bản driver phù hợp với phiên bản Chrome đang sử dụng được tải và cài đặt.
- **selenium.webdriver.chrome.service, selenium.webdriver.chrome.options, selenium.webdriver.common.by, selenium.webdriver.support.ui, selenium.webdriver.support.expected_conditions:** Các module này cung cấp các công cụ cần thiết để cấu hình trình duyệt Chrome, định vị các phần tử HTML trên trang web (ví dụ như các bảng dữ liệu), và thiết lập các điều kiện chờ để đảm bảo rằng các phần tử cần thiết đã được tải trước khi tiến hành trích xuất.
- **bs4 (BeautifulSoup):** Thư viện này giúp phân tích cú pháp của tài liệu HTML mà Selenium đã thu thập được. Nó cho phép em dễ dàng tìm kiếm và trích xuất dữ liệu từ các thẻ HTML, đặc biệt là từ các bảng thống kê.
- **io.StringIO:** Thư viện này cho phép xử lý một chuỗi văn bản như một file. Em đã sử dụng nó để đưa mã HTML của bảng vào hàm pd.read_html của pandas, giúp việc đọc dữ liệu bảng từ HTML trở nên thuận tiện hơn.
- **os, re, uuid:** Các thư viện này cung cấp các công cụ cho các tác vụ liên quan đến hệ điều hành, biểu thức chính quy (để xử lý chuỗi văn bản), và tạo định danh duy nhất. Trong bài code này, re được sử dụng để xử lý dữ liệu tuổi, còn os và uuid có thể được sử dụng cho

các mục đích khác như quản lý file hoặc tạo ID duy nhất (tuy nhiên, không được sử dụng nhiều trong phần trích xuất dữ liệu chính).

3. Quá trình thực hiện và chức năng của code

Bài code hoạt động theo các bước chính sau:

- **Định nghĩa các liên kết và cấu hình:**
    - table_links: Một dictionary chứa các URL của các trang thống kê khác nhau trên FBref (ví dụ: thống kê tiêu chuẩn, sút bóng, chuyền bóng, v.v.) và ID của bảng tương ứng trên mỗi trang.
    - STATS_COLUMNS: Một danh sách các tên cột tiêu chuẩn mà em muốn có trong bộ dữ liệu cuối cùng.
    - COLUMN_MAPPINGS: Một dictionary ánh xạ giữa tên cột trên trang web FBref (có thể khác nhau giữa các bảng) và tên cột tiêu chuẩn trong STATS_COLUMNS. Điều này giúp đảm bảo sự nhất quán trong tên cột của bộ dữ liệu cuối cùng.
- **Thiết lập trình duyệt (setup_driver()):** Hàm này khởi tạo một phiên bản trình duyệt Chrome ở chế độ headless với các tùy chọn được cấu hình để chạy ổn định trong môi trường tự động.
- **Trích xuất dữ liệu từ từng bảng (extract_player_data()):** Hàm này nhận URL và ID của bảng làm đầu vào, sử dụng Selenium để tải trang và BeautifulSoup để phân tích HTML. Sau đó, nó sử dụng pd.read_html để đọc dữ liệu từ bảng vào một DataFrame. Hàm này cũng xử lý các trường hợp bảng có cấu trúc tiêu đề phức tạp (MultiIndex), tìm cột chứa tên cầu thủ, làm sạch dữ liệu (loại bỏ các hàng không cần thiết hoặc trùng lặp), đổi tên cột 'Player', và chọn các cột dữ liệu cần thiết dựa trên loại thống kê.
- **Hàm chính (main()):**
    - Khởi tạo trình duyệt bằng setup_driver().
    - Duyệt qua các liên kết trong table_links và gọi extract_player_data() để thu thập dữ liệu từ mỗi bảng.
    - Hợp nhất (merge) các DataFrame thu được dựa trên cột 'Player' để tạo một DataFrame duy nhất chứa tất cả các thống kê của cầu thủ. Em đã xử lý riêng dữ liệu của thủ môn và sau đó hợp nhất nó với dữ liệu của các cầu thủ khác, đảm bảo rằng chỉ các thống kê liên quan đến thủ môn mới được giữ lại cho những cầu thủ ở vị trí này.

- Lọc ra những cầu thủ đã chơi hơn 90 phút để tập trung vào những cầu thủ có thời gian thi đấu đáng kể.
- Tạo một DataFrame đầu ra (output_df) với các cột được định nghĩa trong STATS_COLUMNS.
- Ánh xạ dữ liệu từ DataFrame đã hợp nhất vào output_df dựa trên COLUMN_MAPPINGS.
- Thực hiện một số bước làm sạch dữ liệu cụ thể cho một số cột (ví dụ: trích xuất mã quốc gia, tuổi).
- Kiểm tra và điền các giá trị bị thiếu bằng 'N/a'.
- Sắp xếp dữ liệu theo tên cầu thủ và lưu DataFrame cuối cùng vào file results.csv với encoding UTF-8-SIG để hỗ trợ các ký tự đặc biệt.
- Đảm bảo rằng trình duyệt được đóng sau khi quá trình thu thập dữ liệu hoàn tất, ngay cả khi có lỗi xảy ra.

4. Kết quả và nhận xét

Sau khi chạy code, một file CSV có tên results.csv sẽ được tạo. File này chứa dữ liệu thống kê của các cầu thủ Premier League từ các bảng khác nhau trên FBref, với các cột được chuẩn hóa theo danh sách STATS_COLUMNS.

Em nhận thấy rằng việc sử dụng Selenium và BeautifulSoup là một phương pháp hiệu quả để thu thập dữ liệu từ các trang web có cấu trúc phức tạp. Việc xử lý các bảng có MultiIndex và việc ánh xạ tên cột là những bước quan trọng để đảm bảo dữ liệu được thu thập chính xác và có cấu trúc rõ ràng.

5. Những khó khăn và hướng phát triển

Trong quá trình thực hiện, em đã gặp một số khó khăn như việc xác định chính xác ID của bảng trên trang web và xử lý các trường hợp cấu trúc HTML của trang web thay đổi. Việc đảm bảo code hoạt động ổn định và có khả năng xử lý lỗi tốt cũng là một thách thức.Đặc biệt là khi dùng request không thể cho ra được thông tin như mong muốn.

BÀI 2: BÁO CÁO QUÁ TRÌNH THU THẬP DỮ LIỆU THỐNG KÊ CẦU THỦ PREMIER LEAGUE 2024-2025 TỪ FBREF

1. Mục tiêu và Tổng quan

Đoạn code này được thiết kế để tự động thu thập dữ liệu thống kê về cầu thủ từ trang web FBref (fbref.com) cho giải đấu Premier League mùa giải 2024-2025. Code sử dụng thư viện Selenium

để điều khiển trình duyệt web (Chrome ở chế độ headless) và BeautifulSoup để phân tích cú pháp HTML của trang web, từ đó trích xuất dữ liệu từ các bảng thống kê khác nhau. Cuối cùng, dữ liệu được xử lý và lưu trữ vào một file CSV có tên results.csv.

2. Các Thư viện và Công cụ Sử dụng

- **pandas:** Được sử dụng để tạo và quản lý DataFrame, là cấu trúc dữ liệu chính để lưu trữ và xử lý dữ liệu thu thập được.
- **selenium:** Một thư viện mạnh mẽ để tự động hóa tương tác với trình duyệt web. Trong code này, Selenium được dùng để mở trang web, chờ cho các bảng dữ liệu được tải hoàn chỉnh (sử dụng WebDriverWait và expected_conditions), và lấy mã nguồn HTML của trang.
- **webdriver_manager:** Hỗ trợ quản lý driver cho trình duyệt Chrome, tự động tải và cài đặt phiên bản driver phù hợp.
- **selenium.webdriver.chrome.service, selenium.webdriver.chrome.options, selenium.webdriver.common.by, selenium.webdriver.support.ui, selenium.webdriver.support.expected_conditions:** Các module con của Selenium cung cấp các lớp và hàm cần thiết để thiết lập trình duyệt, định vị các phần tử trên trang web, và xử lý các điều kiện chờ.
- **bs4 (BeautifulSoup):** Một thư viện Python để phân tích cú pháp các tài liệu HTML và XML. Được sử dụng để duyệt và tìm kiếm các phần tử HTML chứa dữ liệu bảng.
- **io.StringIO:** Một lớp để xử lý dữ liệu chuỗi như một file, giúp pd.read_html có thể đọc dữ liệu HTML trực tiếp từ chuỗi.
- **os:** Cung cấp các hàm tương tác với hệ điều hành, có thể được sử dụng cho các tác vụ liên quan đến file (mặc dù không được sử dụng trực tiếp trong phần trích xuất chính).
- **re:** Thư viện biểu thức chính quy, được sử dụng để thực hiện các thao tác tìm kiếm và thay thế trên chuỗi (ví dụ: tách tuổi).
- **uuid:** Thư viện để tạo các định danh duy nhất (không được sử dụng trực tiếp trong đoạn code chính, có thể là phần còn sót lại hoặc dự định sử dụng).

3. Cấu trúc Code và Quy trình Hoạt động

- **table_links Dictionary:** Định nghĩa một dictionary chứa các liên kết đến các trang thống kê khác nhau trên FBref (ví dụ: Standard Stats, Shooting, Passing) cùng với ID của bảng tương ứng trên trang web.

- **STATS_COLUMNS List:** Một danh sách các tên cột tiêu chuẩn mà code hướng tới để thu thập và đặt tên cho dữ liệu cuối cùng.
- **COLUMN_MAPPINGS Dictionary:** Một dictionary ánh xạ giữa tên cột trên trang web FBref (có thể khác nhau giữa các bảng) và tên cột tiêu chuẩn trong STATS_COLUMNS. Điều này giúp chuẩn hóa tên cột trong DataFrame cuối cùng.
- **setup_driver() Function:** Thiết lập một phiên bản trình duyệt Chrome ở chế độ headless (không hiển thị giao diện người dùng). Các tùy chọn được thêm vào để đảm bảo trình duyệt chạy ổn định trong môi trường tự động.
- **extract_player_data(name, url, table_id, driver) Function:** Hàm này chịu trách nhiệm chính cho việc trích xuất dữ liệu từ một trang thống kê cụ thể.
    - Truy cập URL của trang.
    - Sử dụng WebDriverWait để đợi cho bảng dữ liệu với ID table_id xuất hiện trên trang.
    - Sử dụng BeautifulSoup để phân tích mã nguồn HTML của trang.
    - Tìm bảng dữ liệu cần thiết.
    - Sử dụng pd.read_html để đọc dữ liệu từ bảng HTML vào một DataFrame.
    - Xử lý các cột có cấu trúc MultiIndex (nhiều cấp độ tiêu đề cột) bằng cách kết hợp các cấp độ để tạo tên cột mới.
    - Tìm cột chứa tên cầu thủ (dựa trên từ khóa 'player').
    - Làm sạch dữ liệu bằng cách loại bỏ các hàng bị thiếu tên cầu thủ và các hàng trùng lặp.
    - Đổi tên cột cầu thủ thành 'Player'.
    - Chọn các cột cần thiết dựa trên loại bảng (ví dụ: 'Standard Stats' có các cột khác với 'Shooting').
    - Trả về DataFrame chứa dữ liệu đã trích xuất.
- **main() Function:** Hàm chính điều khiển toàn bộ quá trình.
    - Gọi setup_driver() để khởi tạo trình duyệt.
    - Duyệt qua các mục trong table_links.
    - Gọi extract_player_data() cho mỗi loại thống kê để lấy dữ liệu.

- Hợp nhất (merge) các DataFrame thu được (trừ DataFrame của thủ môn) dựa trên cột 'Player' để tạo một DataFrame duy nhất chứa tất cả các thống kê của cầu thủ. Sử dụng how='outer' để giữ lại tất cả các cầu thủ từ tất cả các bảng.
- Xử lý riêng DataFrame của thủ môn và sau đó hợp nhất nó với DataFrame chính. Các cột thống kê của thủ môn chỉ được giữ lại nếu vị trí ('Pos') của cầu thủ là 'GK'.
- Lọc ra những cầu thủ đã chơi hơn 90 phút (dựa trên cột 'Playing Time Min').
- Tạo một DataFrame output_df với các cột được định nghĩa trong STATS_COLUMNS, bao gồm cả cột 'Player'.
- Ánh xạ dữ liệu từ DataFrame đã hợp nhất vào output_df dựa trên COLUMN_MAPPINGS.
- Thực hiện một số làm sạch và trích xuất thông tin cụ thể từ các cột (ví dụ: lấy mã quốc gia, tuổi).
- Kiểm tra và xử lý các giá trị bị thiếu bằng cách điền 'N/a'.
- Sắp xếp DataFrame theo tên cầu thủ.
- Lưu DataFrame cuối cùng vào file results.csv với encoding 'utf-8-sig' để hỗ trợ các ký tự đặc biệt.
- Đảm bảo đóng trình duyệt sau khi hoàn thành hoặc nếu có lỗi xảy ra (trong khối finally).

4. Nhận xét và Thảo luận

- **Tính tự động hóa:** Code này tự động hóa quá trình thu thập dữ liệu từ web, giúp tiết kiệm thời gian và công sức so với việc thu thập thủ công.
- **Xử lý đa dạng các bảng:** Code được thiết kế để xử lý nhiều loại bảng thống kê khác nhau từ FBref, mỗi bảng có cấu trúc và tên cột khác nhau.
- **Chuẩn hóa dữ liệu:** Việc sử dụng COLUMN_MAPPINGS giúp chuẩn hóa tên cột, làm cho dữ liệu dễ dàng phân tích hơn sau này.
- **Xử lý lỗi:** Code có các khối try-except để xử lý các tình huống lỗi có thể xảy ra trong quá trình tải trang hoặc tìm kiếm bảng.
- **Lọc cầu thủ:** Việc lọc cầu thủ đã chơi hơn 90 phút có thể hữu ích để tập trung vào những cầu thủ có đóng góp đáng kể.

- **Xử lý dữ liệu thủ môn:** Code có logic riêng để xử lý dữ liệu của thủ môn, đảm bảo chỉ các thống kê liên quan đến thủ môn mới được giữ lại cho những cầu thủ ở vị trí này.
- **Sử dụng Selenium Headless:** Chạy trình duyệt ở chế độ headless giúp giảm tải tài nguyên và phù hợp cho các tác vụ chạy nền hoặc trên server.

BÀI 3: Kính thưa thầy/cô, em xin trình bày báo cáo về bài code mà em đã thực hiện để phân tích dữ liệu thống kê cầu thủ bằng phương pháp phân cụm K-means và giảm chiều dữ liệu bằng PCA.

BÁO CÁO BÀI CODE: PHÂN CỤM CẦU THỦ PREMIER LEAGUE DỰA TRÊN THỐNG KÊ

1. Giới thiệu chung

Bài code này sử dụng các kỹ thuật học máy không giám sát, cụ thể là thuật toán phân cụm K-means và phương pháp giảm chiều dữ liệu PCA (Principal Component Analysis), để phân tích dữ liệu thống kê của các cầu thủ tại giải bóng đá Ngoại hạng Anh. Dữ liệu đầu vào được đọc từ file results.csv, là kết quả của quá trình thu thập dữ liệu đã thực hiện trước đó. Mục tiêu của bài code này là nhóm các cầu thủ có phong cách chơi và đặc điểm thống kê tương tự nhau thành các cụm, đồng thời trực quan hóa các cụm này trong không gian hai chiều để dễ dàng nhận diện.

2. Các thư viện đã sử dụng

- **pandas: Tiếp tục được sử dụng để làm việc với DataFrame, là cấu trúc dữ liệu chính để xử lý dữ liệu thống kê.**
- **numpy: Thư viện toán học số học, được sử dụng cho các phép toán trên mảng và xử lý các giá trị số.**
- **matplotlib.pyplot: Thư viện để tạo các biểu đồ trực quan hóa dữ liệu, bao gồm biểu đồ đường cong khuỷu tay (Elbow Method) và biểu đồ phân tán của các cụm sau khi giảm chiều bằng PCA.**
- **sklearn.cluster.KMeans: Thuật toán phân cụm K-means được sử dụng để nhóm các cầu thủ thành các cụm dựa trên sự tương đồng về các chỉ số thống kê.**
- **sklearn.preprocessing.StandardScaler: Được sử dụng để chuẩn hóa dữ liệu trước khi áp dụng thuật toán K-means. Chuẩn hóa giúp đảm bảo rằng tất cả các đặc trưng (các cột thống kê) đều có đóng góp ngang nhau vào quá trình phân cụm, tránh trường hợp một số đặc trưng có giá trị lớn hơn chi phối kết quả.**

- **sklearn.decomposition.PCA: Phương pháp giảm chiều dữ liệu PCA được sử dụng để giảm số lượng các biến (các cột thống kê) xuống còn hai thành phần chính, giúp trực quan hóa các cụm trong không gian 2D.**
- **sklearn.impute.SimpleImputer: Được sử dụng để xử lý các giá trị bị thiếu (NaN) trong dữ liệu bằng cách thay thế chúng bằng giá trị trung vị của cột tương ứng.**
- **os: Thư viện để tương tác với hệ điều hành, trong code này được sử dụng để đảm bảo thư mục histograms tồn tại để lưu trữ các biểu đồ.**

3. Quá trình thực hiện và chức năng của code

Bài code hoạt động theo các bước chính sau:

- **Load dữ liệu: Đọc dữ liệu từ file results.csv vào DataFrame df.**
- **Chọn cột số: Xác định các cột chứa dữ liệu số để sử dụng cho việc phân cụm (tương tự như trong bài code phân tích thống kê mô tả).**
- **Chuẩn bị dữ liệu cho phân cụm:**
    - **Tạo một bản sao của các cột số để tránh ảnh hưởng đến DataFrame gốc.**
    - **Thay thế các giá trị 'N/a' bằng np.nan để xử lý các giá trị bị thiếu.**
    - **Sử dụng SimpleImputer với chiến lược 'median' để điền các giá trị bị thiếu bằng giá trị trung vị của từng cột.**
    - **Sử dụng StandardScaler để chuẩn hóa dữ liệu, đưa các đặc trưng về cùng một thang đo.**
- **Xác định số lượng cụm tối ưu (Elbow Method):**
    - **Chạy thuật toán K-means với số lượng cụm khác nhau (từ 1 đến 10).**
    - **Tính toán WCSS (Within-Cluster Sum of Squares) cho mỗi số lượng cụm. WCSS đo lường tổng khoảng cách bình phương từ mỗi điểm dữ liệu đến tâm cụm gần nhất của nó.**
    - **Vẽ biểu đồ đường cong khuỷu tay (Elbow Curve) để trực quan hóa sự thay đổi của WCSS theo số lượng cụm. Điểm "khuỷu tay" trên biểu đồ thường được coi là số lượng cụm tối ưu. Biểu đồ này được lưu vào file histograms/elbow_curve.png.**
- **Phân cụm K-means:**

- **Dựa trên biểu đồ khuỷu tay và/hoặc kiến thức về lĩnh vực (ví dụ: các vị trí cầu thủ), chọn số lượng cụm (n_clusters). Trong code này, số lượng cụm được chọn là 4.**
- **Khởi tạo và chạy thuật toán K-means với số lượng cụm đã chọn trên dữ liệu đã chuẩn hóa.**
- **Gán nhãn cụm cho từng cầu thủ dựa trên kết quả phân cụm.**

- **Lưu kết quả phân cụm: Tạo một DataFrame mới chỉ chứa tên cầu thủ và nhãn cụm, sau đó lưu vào file clustering_results.csv.**
- **Giảm chiều dữ liệu bằng PCA:**
  - **Sử dụng PCA để giảm số lượng các đặc trưng xuống còn 2 thành phần chính.**
  - **Tính toán tỷ lệ phương sai được giải thích bởi mỗi thành phần chính.**
  - **Biến đổi dữ liệu đã chuẩn hóa sang không gian 2D của các thành phần chính.**
- **Trực quan hóa các cụm trong không gian 2D:**
  - **Vẽ biểu đồ phân tán của các cầu thủ trong không gian 2D, với mỗi điểm đại diện cho một cầu thủ và màu sắc của điểm thể hiện cụm mà cầu thủ đó thuộc về.**
  - **Thêm chú thích về tỷ lệ phương sai được giải thích bởi mỗi thành phần chính vào nhãn của các trục.**
  - **Lưu biểu đồ này vào file histograms/player_clusters_2d.png.**
- **Phân tích và in nhận xét về các cụm: Đưa ra những nhận xét ban đầu về ý nghĩa có thể có của từng cụm dựa trên kiến thức về bóng đá và các chỉ số thống kê. Ví dụ, một cụm có thể đại diện cho các thủ môn do có các chỉ số đặc trưng như tỷ lệ cứu thua cao.**

4. Kết quả và nhận xét

Sau khi chạy code, hai file hình ảnh sẽ được lưu vào thư mục histograms: elbow_curve.png (biểu đồ khuỷu tay) và player_clusters_2d.png (biểu đồ phân tán các cụm trong không gian 2D). File clustering_results.csv chứa danh sách các cầu thủ và nhãn cụm mà họ được gán.

Biểu đồ khuỷu tay giúp xác định số lượng cụm tối ưu bằng cách tìm điểm mà việc tăng thêm số lượng cụm không còn giảm đáng kể WCSS. Biểu đồ phân tán 2D cho thấy sự phân bố của các cầu thủ trong không gian giảm chiều, với các màu khác

nhau biểu thị các cụm khác nhau. Mặc dù có thể có một số chồng chéo giữa các cụm, nhưng nhìn chung, các cầu thủ có đặc điểm tương tự sẽ có xu hướng tập trung lại với nhau.

Các nhận xét ban đầu về ý nghĩa của các cụm cung cấp một cái nhìn sơ bộ về các vai trò hoặc phong cách chơi khác nhau của các nhóm cầu thủ được xác định bởi thuật toán.

5. Những khó khăn và hướng phát triển

Một trong những khó khăn chính là việc xác định số lượng cụm tối ưu một cách chủ quan dựa trên biểu đồ khuỷu tay và kiến thức về lĩnh vực. Đôi khi, biểu đồ khuỷu tay không có một "khuỷu tay" rõ ràng.

Bài 4: **DỰ ĐOÁN GIÁ TRỊ CHUYỂN NHƯỢNG CẦU THỦ BÓNG ĐÁ**

1. Giới thiệu chung

Bài code này nhằm mục đích xây dựng một mô hình học máy có khả năng dự đoán giá trị chuyển nhượng của các cầu thủ bóng đá dựa trên các chỉ số thống kê về hiệu suất thi đấu của họ. Code bao gồm các bước chính sau: đọc và tiền xử lý dữ liệu thống kê cầu thủ, thu thập giá trị chuyển nhượng thực tế từ trang web FootballTransfers, chuẩn bị dữ liệu để huấn luyện mô hình, xây dựng và đánh giá mô hình hồi quy sử dụng thuật toán Random Forest, và cuối cùng (tùy chọn) dự đoán giá trị chuyển nhượng cho toàn bộ tập dữ liệu.

2. Các thư viện đã sử dụng

- **pandas:** Tiếp tục được sử dụng để đọc, xử lý và thao tác với dữ liệu dạng bảng (DataFrame).
- **numpy:** Thư viện toán học số học, hỗ trợ các phép toán trên mảng và xử lý dữ liệu số.
- **json:** Được sử dụng để làm việc với dữ liệu JSON, đặc biệt là để lưu trữ và đọc dữ liệu cache của giá trị chuyển nhượng đã thu thập.
- **os:** Thư viện để tương tác với hệ điều hành, được sử dụng để kiểm tra sự tồn tại của file cache.
- **logging:** Thư viện để ghi lại các sự kiện xảy ra trong quá trình chạy code, giúp theo dõi và gỡ lỗi.
- **random:** Được sử dụng để chọn ngẫu nhiên user-agent cho trình duyệt web, giúp tránh bị chặn khi thu thập dữ liệu.

- **time:** Thư viện để quản lý thời gian, được sử dụng để thêm độ trễ giữa các lần thu thập dữ liệu từ trang web.
- **selenium:** Tiếp tục được sử dụng để tự động hóa việc tương tác với trình duyệt web Chrome, cho phép thu thập giá trị chuyển nhượng từ trang FootballTransfers.
- **webdriver_manager:** Quản lý driver cho trình duyệt Chrome.
- **selenium.webdriver.chrome.service, selenium.webdriver.chrome.options, selenium.webdriver.common.by, selenium.webdriver.support.ui, selenium.webdriver.support.expected_conditions:** Các module con của Selenium để cấu hình trình duyệt, định vị các phần tử trên trang web và xử lý các điều kiện chờ.
- **sklearn.model_selection.train_test_split, GridSearchCV:** Sử dụng để chia dữ liệu thành tập huấn luyện và tập kiểm tra, và để tìm kiếm các siêu tham số tối ưu cho mô hình Random Forest thông qua cross-validation.
- **sklearn.ensemble.RandomForestRegressor:** Thuật toán hồi quy Random Forest được sử dụng làm mô hình dự đoán giá trị chuyển nhượng.
- **sklearn.preprocessing.StandardScaler:** Chuẩn hóa các đặc trưng số trước khi huấn luyện mô hình.
- **sklearn.impute.SimpleImputer:** Xử lý các giá trị bị thiếu bằng cách thay thế chúng bằng giá trị trung vị.
- **sklearn.metrics.mean_squared_error, r2_score:** Các hàm để đánh giá hiệu suất của mô hình hồi quy.

3. Quá trình thực hiện và chức năng của code

Bài code hoạt động theo các bước sau:

- **Đọc và tiền xử lý dữ liệu (load_data()):** Đọc dữ liệu thống kê cầu thủ từ file results.csv, chuẩn hóa tên cột, chuyển đổi các cột số sang định dạng số, và lọc ra các cầu thủ có số phút thi đấu lớn hơn 900.
- **Thu thập giá trị chuyển nhượng (collect_transfer_values()):**
    - Sử dụng Selenium để mở trình duyệt Chrome (ở chế độ headless).
    - Duyệt qua danh sách các cầu thủ từ DataFrame.
    - Tìm kiếm giá trị chuyển nhượng của từng cầu thủ trên trang web FootballTransfers.

    - Sử dụng cache (lưu trong file scrape_cache.json) để tránh thu thập lại giá trị của những cầu thủ đã được thu thập trước đó.
    - Lưu kết quả thu thập (tên cầu thủ và giá trị chuyển nhượng thô) vào một DataFrame.
    - Hàm scrape_player_value() thực hiện việc tìm kiếm và trích xuất giá trị chuyển nhượng cho một cầu thủ cụ thể.
    - Hàm clean_value() chuyển đổi giá trị chuyển nhượng từ định dạng văn bản (ví dụ: '€10m', '£500k') sang định dạng số (EUR).
- **Chuẩn bị features (prepare_features()):**
    - Hợp nhất DataFrame thống kê với DataFrame giá trị chuyển nhượng dựa trên tên cầu thủ.
    - Điền các giá trị chuyển nhượng bị thiếu bằng giá trị trung vị.
    - Thực hiện feature engineering bằng cách tạo các đặc trưng mới như số bàn thắng và kiến tạo trên 90 phút.
    - Chọn một tập hợp các đặc trưng số và mã hóa one-hot cho cột vị trí ('position').
    - Tạo DataFrame cuối cùng (df_model) chứa các đặc trưng đã xử lý và cột mục tiêu ('transfer_value'), loại bỏ các hàng có giá trị NaN còn lại.
- **Đào tạo và đánh giá mô hình (train_and_evaluate()):**
    - Chia dữ liệu đã chuẩn bị thành tập huấn luyện và tập kiểm tra.
    - Sử dụng SimpleImputer để xử lý các giá trị bị thiếu (nếu có) trong tập huấn luyện.
    - Sử dụng StandardScaler để chuẩn hóa các đặc trưng số trong cả tập huấn luyện và tập kiểm tra.
    - Sử dụng GridSearchCV để tìm kiếm các siêu tham số tối ưu cho mô hình RandomForestRegressor thông qua cross-validation trên tập huấn luyện. Các siêu tham số được thử nghiệm là n_estimators (số lượng cây trong rừng) và max_depth (độ sâu tối đa của cây).
    - Huấn luyện mô hình Random Forest với các siêu tham số tốt nhất tìm được.
    - Dự đoán giá trị chuyển nhượng trên tập kiểm tra và đánh giá hiệu suất của mô hình bằng các metrics như Mean Squared Error (MSE) và R-squared (R2).

- **(Tùy chọn) Dự đoán và lưu kết quả:** Phần này (được comment trong code) có thể được triển khai để dự đoán giá trị chuyển nhượng cho toàn bộ tập dữ liệu và lưu kết quả vào file transfer_predictions.csv.

4. Kết quả và nhận xét

Sau khi chạy code, các thông tin về siêu tham số tốt nhất của mô hình và hiệu suất của mô hình trên tập kiểm tra (MSE và R2) sẽ được in ra. Giá trị MSE cho biết độ lớn trung bình của các lỗi dự đoán, trong khi giá trị R2 cho biết tỷ lệ phương sai của biến mục tiêu có thể được dự đoán từ các đặc trưng. Một giá trị R2 gần 1 cho thấy mô hình phù hợp với dữ liệu tốt hơn.

Bài code này đã trình bày một quy trình cơ bản để xây dựng mô hình dự đoán giá trị chuyển nhượng cầu thủ. Việc sử dụng Random Forest, một thuật toán mạnh mẽ và linh hoạt, cho phép mô hình nắm bắt các mối quan hệ phức tạp giữa các đặc trưng và giá trị chuyển nhượng. Quá trình thu thập dữ liệu giá trị chuyển nhượng từ web là một bước quan trọng để có được nhãn (target variable) cho bài toán học có giám sát này.

5. Những khó khăn và hướng phát triển

Một số khó khăn có thể gặp phải bao gồm:

- **Độ chính xác của dữ liệu thu thập từ web:** Giá trị chuyển nhượng trên các trang web có thể không phải là giá trị chính thức và có thể thay đổi theo thời gian.
- **Tính sẵn có của dữ liệu:** Không phải tất cả các cầu thủ đều có thông tin giá trị chuyển nhượng trên trang web.
- **Lựa chọn đặc trưng:** Việc lựa chọn các đặc trưng phù hợp có ảnh hưởng lớn đến hiệu suất của mô hình. Có thể cần thử nghiệm với nhiều đặc trưng khác nhau hoặc sử dụng các phương pháp lựa chọn đặc trưng.
- **Cân bằng dữ liệu:** Dữ liệu có thể bị lệch, ví dụ như có nhiều cầu thủ có giá trị chuyển nhượng thấp hơn so với cầu thủ có giá trị cao.
- **Thời gian thu thập dữ liệu:** Việc thu thập dữ liệu từ web có thể tốn thời gian và có thể bị ảnh hưởng bởi tốc độ mạng hoặc cấu trúc trang web.